



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT

AUTHORING *of* ADAPTIVE SINGLE-PLAYER  
EDUCATIONAL GAMES

Vom Fachbereich Elektrotechnik und Informationstechnik  
der Technischen Universität Darmstadt  
zur Erlangung des akademischen Grades eines  
Doktor-Ingenieurs (Dr.-Ing.)  
genehmigte Dissertation

von

DIPL.-INFORM. FLORIAN MEHM

Geboren am 29. Mai 1982 in Darmstadt

Vorsitz: Prof. Dr.-Ing. Volker Hinrichsen  
Referent: Prof. Dr.-Ing. Ralf Steinmetz  
Korreferent: Prof. Dr.-Ing. Maic Masuch

Tag der Einreichung: 31. Januar 2013  
Tag der Disputation: 10. Mai 2013

Hochschulkennziffer D17  
Darmstadt 2013

#### BIBLIOGRAPHIC INFORMATION

Dieses Dokument wird bereitgestellt von tuprints, E-Publishing-Service der Technischen Universität Darmstadt.

<http://tuprints.ulb.tu-darmstadt.de>  
[tuprints@ulb.tu-darmstadt.de](mailto:tuprints@ulb.tu-darmstadt.de)

Bitte zitieren Sie dieses Dokument als:

URN: [urn:nbn:de:tuda-tuprints-37218](https://nbn-resolving.org/urn:nbn:de:tuda-tuprints-37218)

URL: <http://tuprints.ulb.tu-darmstadt.de/id/eprint/3721>

In loving memory of Marie Mehm.





## ABSTRACT

---

**D**IGITAL Educational Games, as one of the most important application areas of Serious Games, combine positive properties of digital games, such as strong motivation for players and inherent learning processes, with educational methods and technologies. Adaptive algorithms allow such games to be aligned automatically to the needs of different players, thereby increasing the learning efficacy. However, educational games are among the most complex game production endeavors, since they are often faced with small budget on the one hand and special requirements with impacts on all aspects of game development, from design through programming to asset production, on the other.

Authoring tools have been successfully created and used in fields related to Serious Games and educational games, including e-Learning, multimedia, interactive storytelling and entertainment games. These tools incorporate parts of the production workflows in their respective areas and allow all authors, including non-programmers, to create applications. While it appears beneficial to create authoring tools for educational games, we find that authoring tools for educational games have to account for the higher complexity and interactivity of games compared to other forms of multimedia and that they have not been researched thoroughly in the past.

These challenges are addressed in this thesis by presenting a concept for an authoring tool for adaptive educational single-player games that accounts for the specifics of educational game development. Major results are an educational game description model, concepts for adaptive control of educational games and author support mechanisms specifically for adaptive educational game authoring. These concepts are implemented in the authoring tool “StoryTec”, which is validated in the course of a set of evaluation studies. The novel features of StoryTec include the specific support for adaptive educational games, a concept for structural and interaction templates shown to increase the efficiency and effectiveness of the authoring tool, as well as the support for collaborative work. It builds the foundation for a number of current and future research and development projects, including the extension towards authoring of multi-player games, and is tested and used by over 120 members of an open community.



## KURZFASSUNG

---

**L**ERNSPIELE, als eine Form von Serious Games, nutzen die positiven Eigenschaften von Spielen, unter anderem die allgegenwärtigen Lernprozesse und motivierenden Effekte, um Lernende effektiv zu unterstützen. Während diese Kombination einen hohen Nutzen verspricht, ist die Entwicklung von Lernspielen sehr komplex im Vergleich zu reinen Unterhaltungs-Spielen. Dies ergibt sich aus den oftmals geringeren Budgets auf der einen Seite und zusätzlichen Anforderungen auf der anderen. Diese Anforderungen resultieren aus der Kombination von Spieleentwicklung und Lerninhalten, und äußern sich in Vorgaben für alle Schritte der Spielerstellung, von speziell angepassten Game Design und Inhalten bis zu der technischen Umsetzung.

Autorensysteme haben sich in verwandten Gebieten wie dem E-Learning, Multimedia-Produktion, interaktiven Geschichten oder Unterhaltungs-Spielen als nützlich erwiesen. Diese Systeme bilden Aspekte der Produktionsketten in den jeweiligen Gebieten ab und unterstützen Autoren, also oftmals Nicht-Programmierer, in der Erstellung von interaktiven Anwendungen. Allerdings ist festzustellen, daß Autorensysteme für Spiele ein höheres Maß an Komplexität und Interaktivität unterstützen müssen als Autorensysteme in angrenzenden Gebieten.

Diese Herausforderung wird in dieser Arbeit adressiert mittels eines Konzeptes für ein Autorensystem für adaptive Lernspiele, das die besonderen Anforderungen dieser SpieleGattung unterstützt. Hauptergebnisse sind ein Modell für die Beschreibung von Lernspielen, Konzepte für die adaptive Kontrolle von Lernspielen sowie Mechanismen zur spezifischen Unterstützung von Autoren bei der Erstellung von adaptiven Lernspielen. Diese Konzepte sind exemplarisch anhand des Autorensystems "StoryTec" umgesetzt, das in einer Reihe von Studien evaluiert wurde. Die neuartigen Eigenschaften von StoryTec sind die spezifische Unterstützung der Erstellung von adaptiven Lernspielen, ein Konzept für Struktur- und Interaktions-Vorlagen durch die eine Steigerung der Effizienz und Effektivität des Autorensystems gezeigt werden konnte. Weiterhin unterstützt das Autorensystem die kollaborative Erstellung von Lernspielen.

Das Autorensystem wird als zentrale Komponente in einer Reihe von Forschungsprojekten eingesetzt, in deren Rahmen auch die Unterstützung für Mehrspieler-Spiele erforscht wird, und wird von über 120 Personen im Rahmen einer "Open Community" verwendet und getestet.

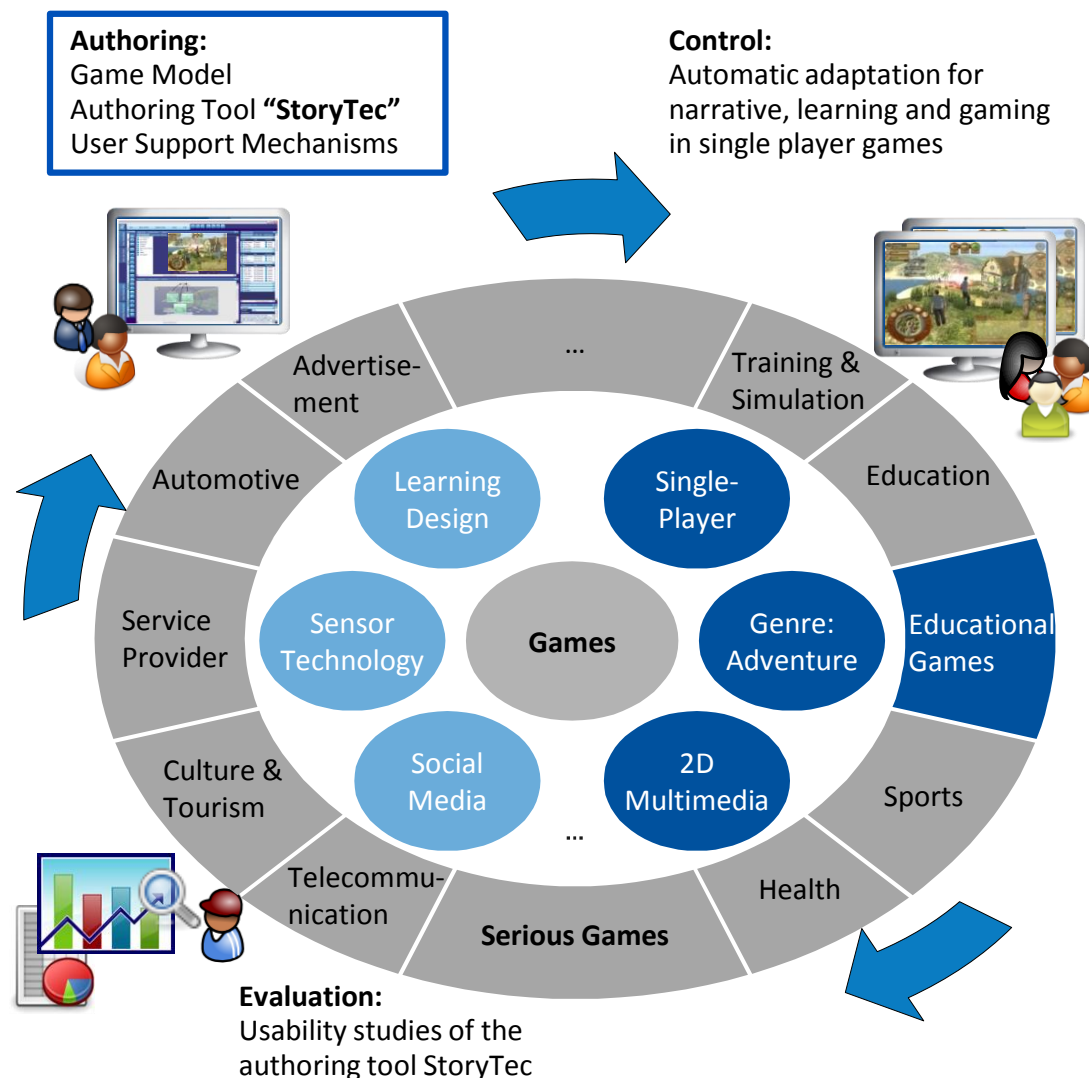


## CONTEXT OF THE THESIS

---

**S**ERIOUS GAMES are games that are created for a primary purpose beyond pure entertainment. They draw their effectiveness from inherent positive properties of games, including the broad appeal of games and high motivation. Three main aspects are relevant: Authoring, referring to the creation of games; Control, including mechanisms for increasing the game's impact during runtime by personalization and adaptation; Evaluation for the measurement of game effects.

In this thesis, an authoring tool for the Serious Game-genre of single-player adaptive educational games is presented. The main contributions are in the first sector of authoring, with related concepts for control and evaluation serving to provide a runtime environment for adaptive educational games.





## ACKNOWLEDGMENTS

---

»Many people die with their music still in them. Why is this so? Too often it is because they are always getting ready to live. Before they know it, time runs out.«

— Oliver Wendell Holmes

FIRST OF ALL, I thank my supervisor, Prof. Dr.-Ing. Ralf Steinmetz, for providing a positive physical and social environment that is perfectly suited for taking on the task of finishing a dissertation. Next, I would like to express my gratitude to all the previous and current members of the Serious Games group, for fruitful discussions, many interesting conversations about games and for the support during the preparation of this thesis. I would like to especially thank my group leader, Dr. Stefan Göbel, for the continuous support beginning with my first contact with interactive storytelling and serious games during my basic studies all the way to the defense of this thesis. Further thanks go to all members of the Multimedia Communications Lab for the good collaboration and the countless cakes! I would also like to thank the members of the Digital Storytelling group at the former Computer Graphics Center for the opportunity for the first step on the path to this thesis and for providing the environment for the foundations of my work. Thanks also go to our partners, especially Sabrina Vogt, who provided many conceptual and practical inputs to the development of StoryTec, and the brothers Konrad and Musal of KTX Software. Finally, I express my gratitude to all of my friends and my family, especially my parents, who supported me throughout my work in many ways.





## CONTENTS

---

|       |  |    |
|-------|--|----|
| 1     | INTRODUCTION .....                             | 1  |
| 1.1   | Motivation .....                               | 1  |
| 1.2   | Context .....                                  | 2  |
| 1.3   | Contributions .....                            | 3  |
| 1.4   | Thesis Organization .....                      | 4  |
| 2     | RELATED WORK .....                             | 7  |
| 2.1   | Educational Games .....                        | 7  |
| 2.1.1 | Educational Adventure Games .....              | 9  |
| 2.2   | Adaptive Systems .....                         | 10 |
| 2.2.1 | Adaptive E-Learning .....                      | 11 |
| 2.2.2 | Adaptive Storytelling .....                    | 12 |
| 2.2.3 | Adaptive Educational Games .....               | 13 |
| 2.3   | Authoring Tools .....                          | 17 |
| 2.3.1 | E-Learning Authoring Tools .....               | 18 |
| 2.3.2 | Interactive Storytelling Authoring Tools ..... | 18 |
| 2.3.3 | Educational Game Authoring Tools .....         | 20 |
| 2.4   | Game Models .....                              | 22 |
| 2.4.1 | E-Learning Models .....                        | 24 |
| 2.4.2 | Interactive Storytelling Models .....          | 25 |
| 2.4.3 | Game Models .....                              | 28 |
| 2.5   | Authoring Support for Non-Programmers .....    | 29 |
| 2.5.1 | Natural Language Programming .....             | 30 |
| 2.5.2 | Visual Programming .....                       | 30 |
| 2.6   | Development of Educational Games .....         | 31 |
| 2.6.1 | Digital Game Development .....                 | 32 |
| 2.6.2 | Educational Game Development .....             | 33 |
| 2.7   | Conclusion .....                               | 36 |
| 3     | CONCEPTUALIZATION AND APPROACH .....           | 39 |
| 3.1   | Application Scenarios .....                    | 39 |
| 3.2   | User Requirements Study .....                  | 41 |
| 3.3   | Requirements Analysis .....                    | 43 |
| 3.4   | Conceptualization .....                        | 46 |
| 3.5   | Conclusion .....                               | 50 |
| 4     | EDUCATIONAL GAME MODEL .....                   | 53 |
| 4.1   | Game Structure Model .....                     | 53 |
| 4.1.1 | Scene .....                                    | 53 |
| 4.1.2 | Object .....                                   | 54 |
| 4.1.3 | Transition .....                               | 55 |
| 4.1.4 | Parameters .....                               | 55 |
| 4.2   | Game Logic Model .....                         | 56 |
| 4.2.1 | Action Set .....                               | 57 |

|       |  |     |
|-------|--|-----|
| 4.2.2 | Stimulus .....   | 57  |
| 4.3   | Runtime Model Execution .....                          | 59  |
| 4.4   | Conclusion .....                                       | 60  |
| 5     | ADAPTIVITY IN EDUCATIONAL GAMES .....                  | 61  |
| 5.1   | Narrative Model .....                                  | 64  |
| 5.2   | Player Model .....                                     | 65  |
| 5.2.1 | Personality-Oriented Models .....                      | 65  |
| 5.2.2 | Player-type based Models .....                         | 66  |
| 5.3   | Learning Model .....                                   | 69  |
| 5.4   | Narrative Game-Based Learning Objects .....            | 71  |
| 5.4.1 | Game Event Interpretation .....                        | 73  |
| 5.4.2 | Model Update .....                                     | 74  |
| 5.4.3 | Adaptive Choices .....                                 | 79  |
| 5.4.4 | Integration into the Authoring Process .....           | 80  |
| 5.5   | Conclusion .....                                       | 81  |
| 6     | AUTHOR SUPPORT .....                                   | 83  |
| 6.1   | Template-based Authoring .....                         | 83  |
| 6.1.1 | Structural Templates .....                             | 83  |
| 6.1.2 | Interaction Templates .....                            | 84  |
| 6.2   | Collaborative Authoring .....                          | 93  |
| 6.3   | Iterative Authoring .....                              | 94  |
| 6.4   | Model Checking .....                                   | 96  |
| 6.5   | Conclusion .....                                       | 101 |
| 7     | IMPLEMENTATION .....                                   | 103 |
| 7.1   | Authoring Tool Implementation: StoryTec .....          | 103 |
| 7.1.1 | Story Editor .....                                     | 104 |
| 7.1.2 | Stage Editor .....                                     | 105 |
| 7.1.3 | Objects Browser .....                                  | 106 |
| 7.1.4 | Property Editor .....                                  | 106 |
| 7.1.5 | Skill Tree Editor .....                                | 107 |
| 7.1.6 | ActionSet and Condition Editor .....                   | 107 |
| 7.2   | Implementation of Adaptive Features .....              | 109 |
| 7.3   | Implementation of Interaction Templates .....          | 111 |
| 7.4   | Authoring Process .....                                | 112 |
| 7.5   | Runtime Environment Architecture .....                 | 114 |
| 7.6   | Rapid Prototyping Tool Implementation: StoryPlay ..... | 119 |
| 7.6.1 | Variables .....  | 120 |
| 7.6.2 | History .....  | 121 |
| 7.6.3 | Narrative Context .....                                | 121 |
| 7.6.4 | Player Model and Gaming Context .....                  | 122 |
| 7.6.5 | Skills and Knowledge Space .....                       | 122 |
| 7.7   | Conclusion .....                                       | 123 |
| 8     | EVALUATION .....                                       | 127 |
| 8.1   | Initial Usability Study .....                          | 128 |
| 8.2   | Follow-up Usability Study .....                        | 130 |

|       |  |     |
|-------|--|-----|
| 8.3   | Focus Group Study with Game Developers .....             | 131 |
| 8.4   | Comparative Study with e-Adventure .....                 | 132 |
| 8.5   | Conclusion .....   | 137 |
| 9     | CONCLUSION AND OUTLOOK .....                             | 139 |
|       | BIBLIOGRAPHY .....                                       | 143 |
|       | LIST OF FIGURES .....                                    | 157 |
|       | LIST OF TABLES .....                                     | 160 |
|       | LIST OF ACRONYMS .....                                   | 161 |
| A     | PROJECTS REALIZED WITH STORYTEC .....                    | 163 |
| A.1   | StoryTec Open Community .....                            | 163 |
| A.2   | 8oDays .....   | 163 |
| A.2.1 | Immersive 3D Demonstrator .....                          | 165 |
| A.2.2 | Rapid Prototyping and Evaluation Platform “Bat Cave” ... | 166 |
| A.3   | Geograficus .....  | 167 |
| A.4   | Wintermute .....   | 168 |
| A.5   | PEDALE .....   | 170 |
| A.6   | Der Chaos-Fluch: Darmstadt im Bann des Zauberers .....   | 170 |
| A.7   | Der Wechsel .....  | 171 |
| A.8   | TechCraft .....  | 172 |
| A.9   | Virtual Sports Teacher .....                             | 174 |
| A.10  | Motivotion 60+ .....                                     | 174 |
| B     | QUESTIONNAIRES .....                                     | 179 |
| B.1   | Initial Usability Study Questionnaire .....              | 180 |
| B.2   | Follow-Up Usability Study Questionnaire .....            | 182 |
| B.3   | Comparison Study Questionnaire .....                     | 187 |
| B.4   | Comparison Study Task Description .....                  | 191 |
| C     | AUTHOR’S PUBLICATIONS .....                              | 195 |
| C.1   | Main Publications .....                                  | 195 |
| C.2   | Co-authored Publications .....                           | 196 |
| D     | CURRICULUM VITÆ .....                                    | 199 |
| E     | ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG .....            | 201 |

## INTRODUCTION

---

»We do not stop playing because we grow old, we grow old because we stop playing.«

— George Bernard Shaw (1856 – 1950)

**T**HIS chapter provides a general overview of this thesis. It consists of the motivation for the addressed topics, a summary of the scientific contributions and an outline of the remaining chapters.

### 1.1 MOTIVATION

Several authors in the past decades proposed the use of digital games in order to support players in various “serious” purposes. One of the first and major developments that has grown from this is the field of digital educational games (DEG) which uses games to impart knowledge and train users [123] [114].

Digital Games are well suited for educational purposes by leveraging strengths of games in general: high acceptance (especially among children and adolescents), high levels of motivation and immersion as well as intrinsic learning processes. Learning in games is inherent by design since players are required to learn the game in order to advance. Eck [39] comments:

“The second factor involves today’s ‘Net Generation’, or ‘digital natives’, who have become disengaged with traditional instruction. They require multiple streams of information, prefer inductive reasoning, want frequent and quick interactions with content, and have exceptional visual literacy skills – characteristics that are all matched well with Digital Game-based Learning (DGBL).”

The effects of educational games can be augmented by personalizing games towards users by means of adaptivity, for example concerning play preferences, learning styles or previous knowledge. Games supporting adaptivity assess the player and choose appropriate game elements based on an internal model of the player. This approach is commonly used in e-Learning applications and can lead to a higher effectiveness of the resulting application [31]. However, adaptivity in digital games has not seen widespread use so far. The associated increase in production costs, due to the required amount of additional content to facilitate adaptation and the higher complexity of creating an adaptive system, can account for this.

Digital Games in general are constituted of a large number of aspects, including technical components such as graphics engines for rendering visuals and game engines as well as large amounts of content (graphics, sounds, animations but also narrative as well as scripts which control the game). Games are often required to be playable on a multitude of platforms, requiring portability to be supported. Heterogeneous development teams are required in which game designers, technicians, artists, writers and other specialists have to collaborate effectively in order to create a

game. Due to this collaborative work, common challenges of interdisciplinary work can lead to delays in development.

In the development of an educational game, all team members have to account for the educational goal of the game, towards which all of its aspects have to be aligned. Domain experts have to be included, adding further collaboration issues and overhead. Furthermore, the development of an educational game typically has to be carried out with a smaller budget than those available to pure entertainment games.

The main challenges in the production of educational games identified so far (complexity, collaboration overhead, limited budget) can be supported by providing an authoring tool for such games. To underline this statement, we turn towards other fields in which authoring tools are common: multimedia, e-Learning and interactive storytelling. An authoring tool commonly abstracts from an underlying technical system and provides the system's functionality to a user in a simplified fashion, often empowering users without knowledge of the technological basis (e.g. programming languages) to create applications running on the system nevertheless. For example, an author using an e-Learning authoring tool can create an interactive web-based training application without knowledge of web technologies or programming languages.

Furthermore, authoring tools provide structured authoring, i.e. a process model which authors follow while creating an application. This process model can also include support for the collaboration between different users.

Therefore, an authoring tool for educational games promises to be capable of addressing the identified challenges. Both the complexity of educational game development, especially in the case of adaptive games, as well as the collaboration overhead during development can be reduced by providing an authoring tool which supports educational game development. By reducing overhead, use of the authoring tool can help meet limited budgets. Especially in cases where authors lack key technical skills such as programming, an authoring tool can empower them to create games. However, due to the high immersion, interactivity, non-linearity and tightly interwoven logic between elements in games, the concepts from authoring tools in related fields have to be adapted to be usable for educational games, too.

## 1.2 CONTEXT

This thesis is prepared in the context of research on Serious Games. Three relevant areas are important in this context: authoring, control and evaluation. Our focus lies on authoring, providing the concept for an authoring tool for educational games. The area of control is addressed by establishing a platform for adaptive educational games created with the authoring tool to be played on. Concerning evaluation, we provide evaluation methodologies for authoring tools and apply them to the authoring tool elaborated here.

In the research of Serious Games, additional dimensions are relevant. One is the separation between single- and multi-player games. Multi-player games, while promising to increase learning effects and teach soft skills along with educational content, require specialized concepts such as synchronization of players and support for group adaptation, which are outside of the scope of this thesis. Another relevant

dimension is the game genre, determining the possible uses of a Serious Game. We choose the game genre of adventure games as the main genre in our work, as it is a common and useful choice in for educational games. Concerning content, game authoring can build upon normal multimedia content, multimodal content or generated content, among others. In order to retain simplicity in the authoring process, we focus on 2D multimedia content. Finally, Serious Games can be built upon specific technologies in a synergistic way, for example communication in networks as is peer-to-peer games. The foundation of the authoring tool concept is content modeling as found in domain-driven development, along with adaptive technologies.

### 1.3 CONTRIBUTIONS

The overall contribution of this thesis is the concept of an authoring tool for adaptive educational single-player games that can be used in various contexts, ranging from single authors creating a game without outside assistance to game development teams working collaboratively.

**EDUCATIONAL GAME DESCRIPTION MODEL** Authoring tools provide structures in order to shield users from underlying basic technologies and to provide a simple authoring process. The necessary precondition for achieving both goals is a model with which structures in the authoring domain can be described. The foundations for such a model can be found in existing modeling approaches in games and game creation tools. We derive a game description model for educational games from the existing approaches which can be used to describe games from a number of genres by including both structural descriptions as well as representations of logical structures.

**ADAPTATION MODEL** We extend the description model for educational games to allow the description of adaptive games. This extension towards adaptivity regarding learning, playing and narration allows games to be adapted at runtime in order to increase motivation, learning efficacy and enjoyment.

**AUTHORING PROCESS MODEL** In order to support collaborative work in game development, we analyze and summarize the current processes in the development of entertainment games and educational games in order to derive a process model. The information about the workflows of various user groups is essential for the conceptualization of an authoring tool. For example, an educator will typically be interested in the instructional design of the game, whereas the author in charge of the narrative will be working with the story model of the game. Furthermore, in some cases, such as productions with limited budget or single users, authors will work in fields that they have not been trained in. Therefore, the process model is incorporated into the authoring tool in such a fashion that unfamiliar users are guided and provided with a best practice workflow. For example, an author untrained in a game-related field should be provided with a template to assist in the given task.

**AUTHORING TOOL** The concepts of this thesis are exemplarily implemented and validated as the authoring tool “StoryTec” for adaptive educational games. It

allows collaborative work and enables non-programmers to work on educational games, thereby addressing the requirements indicated here. In order to demonstrate the approach, a runtime component for interpreting the game description model is created and used in several player applications, controlling games based on the description by authors created in the authoring tool.

**USER SUPPORT MECHANISMS** We provide a set of user support mechanisms that specifically support authors of adaptive educational games in their work. Central importance is given to templates, which include structural templates as well as interaction templates that play a crucial role in non-programmer support and in ensuring portability of the created games by encapsulating reusable game elements. Further mechanisms are rapid prototyping for iterative design of games, semantic examinations by means of model checking and support for collaborative authoring.

**EVALUATION** We describe the study design and results of validation studies of the authoring tool StoryTec. Two usability studies show the basic applicability of StoryTec for the task of game authoring and indicate that its usability is comparable to common office tools. A focus group study with game developers indicates that the concept of StoryTec can be applied in game development. In a study comparing StoryTec with the related authoring tool e-Adventure, the efficiency and effectiveness of StoryTec is demonstrated.

#### 1.4 THESIS ORGANIZATION

The chapters of this thesis are structured as described in the following:

Chapter 1 introduces the topic of this thesis, namely of a concept for authoring tools to be used in the creation of adaptive educational single-player games, especially without the need for knowledge of a programming language. The main contributions of the thesis are presented, consisting of a concept combining several methods for the creation of adaptive games, the implementation of this concept in the authoring tool StoryTec and an evaluation methodology for authoring tools which is used to evaluate the performance of StoryTec.

Chapter 2 assembles the state of the art in the fields of authoring tools in general and various aspects of game development that are utilized in later chapters. The general approach underlying authoring tools is demonstrated based on an overview of authoring tools from the disciplines multimedia, e-Learning and games. The current approaches towards adaptive systems and adaptive educational games are presented. Since they are used to meet the major requirement of supporting non-programmers, the concepts of visual and natural language programming are analyzed. This chapter concludes with an overview of the current process for the creation of digital educational games.

Chapter 3 serves as the linking element between the state of the art and the authoring tool concept. It provides a problem analysis, resulting in a set of application scenarios and a list of user requirements that apply to an authoring tool for adap-

tive educational games. The analysis of the application scenarios includes the roles of team members in the game development process, their individual tasks and required skills as well as the communication and interaction between them. Drawing upon these application scenarios and the requirements analysis along with the related work analyzed in chapter 2, the concept for an authoring tool to be used in the creation of adaptive educational games by non-professionals is introduced.

Chapter 4 focuses on the educational game description model that is used to capture and encode games in the authoring tool. After comparing the modeling methods found in multimedia, interactive storytelling, game authoring tools and model-driven software development approaches, we decide on basing the model on those found in interactive storytelling. The structural aspect of the model consisting of an interlinked structure of scenes as well as the logical aspect including commands and events are presented.

Chapter 5 addresses adaptivity in educational games by providing models for adaptation based on narration, learning and playing. These adaptation dimensions are combined in the concept of Narrative Game-Based Learning Objects (NGLOB), which are realized as an extension to the game description model. We show the process of deriving a method for updating the NGLOB model during play, using the results of a simulation study to indicate a suitable update algorithm.

Chapter 6 builds upon the model of game structure and content elaborated in the previous chapter in order to discuss a number of means for providing assistance to authors. The concepts of structural templates for providing common structures to authors and interaction templates for re-usable encapsulations of game elements are introduced. In order to support collaborative authoring, author roles and visibility options of game model elements depending on the roles are defined. Support for iterative authoring is realized by including a dedicated rapid prototyping tool in the authoring tool concept. The structure of the game model is used in the form of model checking for assisting authors in finding semantic errors and potential problems in games.

Chapter 7 summarizes the implementation of the authoring tool concept elaborated in the previous chapters in the form of StoryTec. The major software components of StoryTec and their relationships are presented, along with a mapping to aspects of the concept and their practical realization in the actual authoring tool. The description of the software also includes the player components “StoryPlay”, to be used as a rapid prototyping and evaluation tool, and “StoryPublish”, which utilizes the structured authoring approach of StoryTec based on the game description model to allow cross-platform publishing on a number of diverse hardware platforms.

Chapter 8 is focused on the performance evaluation of the concept and its implementation in StoryTec. First, an overview of possible evaluation methodologies for authoring tools is provided. This overview is followed by the application of these evaluation methods in order to measure various effects of StoryTec. The results of studies involving two usability studies with StoryTec, a focus group study with members of a professional game development studio and a comparison study of StoryTec



with the related authoring tool e-Adventure are presented. The main results of these studies, underlining the applicability of the concept in game authoring and the advantages of the interaction template concept, are summarized in this chapter.

Chapter 9 concludes the thesis by a summary of the major results of conceptualization, implementation and evaluation of the authoring tool StoryTec. This is followed by an outlook of possible future work, including the incorporation of procedural generation of content, novel authoring methods such as in-game editing and support for multiplayer games.

## RELATED WORK

---

**I**N this chapter, we provide an overview of related work relevant to establishing the background and laying the foundation for the authoring tool concept to be developed in the following chapters. The related work is differentiated into the fields of educational games, research in adaptivity, authoring tools as well as the supporting fields of visual and natural language programming languages. The chapter is concluded by an analysis of the currently common processes in educational game development. The following chapters describing the concept of an authoring tool for adaptive educational games combine the separate ideas from this related work in a novel way.

### 2.1 EDUCATIONAL GAMES

Educational Games are in use at various public and private institutions in a multitude of fields. They are a sub-category of Serious Games, which target fields such as health, school, politics, ethics, military, corporate simulations and more. The high demand and potential of Serious Games in general and educational games in special can be witnessed in many publications and events, including the Serious Games Conference in conjunction with CeBIT<sup>1</sup> or the Game Days<sup>2</sup>.

The usefulness of educational games is backed by a multitude of accounts, including those by Eck [39], Shaffer [127], Prensky [114], [53] and Gee [46]. Marc Prensky in his foundational book on DGBL discusses the cognitive effects of modern media on learners writes [113, chapter 2]:

“So, in the end, it is all these cognitive differences, resulting from years of ‘new media socialization’ and profoundly affecting and changing the generations’ learning styles and abilities, that cry out for new approaches to learning for the Games Generation with a better ‘fit’. And while certainly not the only way, computer games and video games provide one of the few structures we currently have that is capable of meeting many of the Games Generation’s changing learning needs and requirements. This is the key reason why Digital Game-Based Learning is beginning to emerge and thrive.”

The goal of educational games lies in realizing knowledge transfer that is fun and motivating for players. They draw their strengths from several factors, among them the popularity of computer games, especially for younger users. Apart from the popularity of games, the idea of educational games has also been underpinned by the changes in the perception of learning due to computers [127]. One often-cited motivation for educational games is flow theory as established by Csikszentmihalyi

---

<sup>1</sup> <http://www.biu-online.de/en/association/projects/serious-games-conference.html>, last visited on January 24, 2013

<sup>2</sup> <http://www.gamedays2012.de>, last visited on January 24, 2013

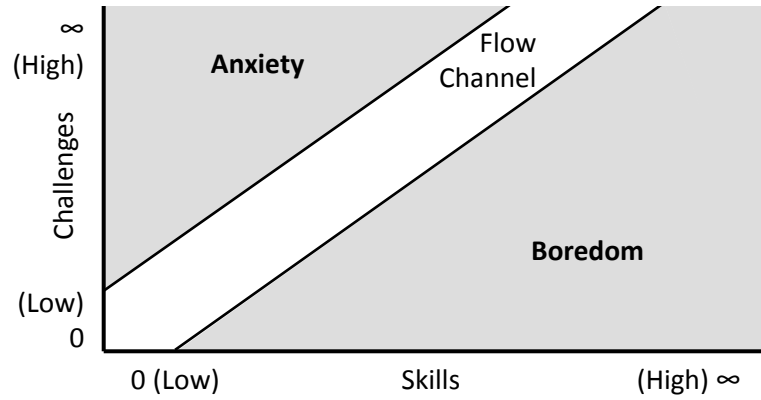


Figure 1: The flow channel as described by Csikszentmihalyi [33], in which learners are neither overstrained nor bored. Source: adapted from [33].

[33], which describes a state of flow in which no conscious effort is required by the person acting and in which the perception of time is affected. Figure 1 visualizes this state, in which players of educational games should ideally be during play. Since game design is often built upon keeping players in this zone (see [126, chapter 24]), a transfer of the positive effects to the learning part of the game is expected.

Initially, the combination of learning and player appears to be a conundrum similar to the one found in the term “Serious Game”. One could argue that a focus on incorporating motivational game play elements might deter from the learning objectives of a game. Conversely, when the focus is laid on the learning objectives, the game aspect might become uninteresting and lead to a higher drop-out rate [74]. The vision of educational games is the synergy between playing and learning, with both aspects acting in an equilibrium [72].

In practice, the creation of educational games is still a relatively young field. Many examples still are “drill and practice” games, which are more akin to quizzes than games and therefore do not benefit from the synergistic effect of combining games and learning. Papert [106] described this phenomenon as “Shavian reversals - offspring that keep the bad features of each parent and lose the good ones”.

Especially adolescents are discouraged by systems that are only focused on knowledge transfer [3]. Furthermore, the learning content of educational games is often fixed and does not take into account the learning preferences and previous knowledge of players [123]. The promise of adaptive educational games is the mitigation of this shortcoming by striving towards providing an optimal mixture of learning and player.

In the following, we differentiate between entertainment games which are not created with a primary purpose beyond entertainment. In contrast, we refer to educational games as those games that have been developed with a focus on education. The spirit of this separation lies in distinguishing games by the purpose they were developed for or by the purpose they are used for mainly in practice. Therefore, games that have been clearly created for entertainment purposes but which can be used in practice to promote a secondary purpose, are counted in this thesis as entertainment games. An example would be a commercial role-playing game that incidentally teaches soft skills such as cooperation and leadership capabilities.

Two further terms that are used in this thesis are “gameplay” and “game mechanics”. The two terms are often used synonymously; however, they can be used to refer to two different game phenomena. A definition for “game mechanics” is provided by Sicart [132] as “[...] methods invoked by agents, designed for interaction with the game state”. The definition shows that game mechanics are related to the logic and computations of digital games, which can be seen as metaphors for rules in non-digital games.

“Gameplay” is used here to refer to the experience players have while interacting with a game. It is therefore linked to game experience as described by Lindley et al. [85]. Factors that influence the gameplay include visuals, sound, narrative, game content and control, difficulty and freedom of choice. As Salen and Zimmerman [126, p. 127ff] point out, similar game mechanics coupled with varying gameplay factors such as background story or visual style can lead to different perception of the game by players.

A well-known example of an educational game is “Global Conflicts: Palestine” [19], intended to educate about the Middle East conflict. In this game, the player assumes the role of a reporter assembling a newspaper article for an Israeli, Palestinian or European (neutral) newspaper. In order to accomplish this, the player questions experts and witnesses, for example the witnesses of a suicide bombing. Players have to judge the validity of statements by the involved parties and can influence the support by them by choosing a style of writing. By interacting in this conflict in an immersive way, the players learn the motivations and problems of the involved parties.

Apart from games produced as educational games in the first place, several entertainment games have been used or researched in the context of learning. A popular example is “Minecraft”, a game using procedurally created 3D worlds made of blocks in which players can create structures freely from the available resources. Apart from creating elaborate worlds and machines, users have created educational scenarios using Minecraft [38].

### 2.1.1 Educational Adventure Games

Educational games can be created in many different game genres effectively. However, some genres are more suitable for learning than others. Gros [52] provides a list of seven genres that are suitable for educational games. We highlight the genre of educational adventure games here, as it can be seen as one of the most suitable for educational games. This genre is defined by the focus on slow gameplay and fewer action-laden or time-restricted sequences than games from other genres. Educators have chosen this genre to build educational games both during the initial era of commercial success of adventure games (cf. [23]) as well as during recent times (cf. [93]). Other examples are the cultural heritage adventure game “Ataiyal Legend” [54] or the action-adventure game “Ludwig” for teaching physics [157].

The German educational game studio Braingame and publisher Heureka produced a commercially released series of educational adventure games including

“Physicus”<sup>3</sup>, “Informaticus”<sup>4</sup> and “Geograficus”<sup>5</sup> (teaching about physics, computer science and geography, respectively). The genre of first-person adventure game was chosen, in which player use pointing and clicking to move a first-person camera through a pre-rendered environment shown in 2D images.

Adventure games commonly use strong narratives to embed puzzles [36]. This means can be used to transport educational content, which is embedded in the game world and puzzles. Without time limits, the players can discover this knowledge on their own schedule. Similar to historical novels, the learning content can be embedded in the narrative of the game (cf. [16]).

From a production perspective, gamers have lower expectations concerning graphics or visual effects in adventure games as compared to other commercial game genres. While consumers expect other genres to push the boundaries of the hardware possibilities, adventure games can be successful without cutting edge game techniques. Furthermore, the strong conventions in the genre of adventure games lead to many games sharing similar game mechanics and interface elements, which can reduce the design effort for these elements. These two aspects lead to a reduction in costs, which is a necessary factor to account for in many educational game productions.

## 2.2 ADAPTIVE SYSTEMS

In this section, the related work of adaptive systems is described. After a general overview of adaptive software, we analyze the adaptive approaches in e-Learning, in educational games and the in the related field of adaptive interactive storytelling.

Adaptive software in general is characterized as a system that shows behavior that varies based on the users that are currently using it, based on properties, preferences, goals or capabilities. This is done with the goal of ensuring that a user remains in the flow zone as postulated by Csikszentmihalyi, neither overstraining nor boring a user and optimizing the user’s productivity.

In order to provide this adaptation, the properties of users have to be modeled in a process which monitors their actions and updates the model accordingly. The umbrella term for all approaches that use this concept is user modeling.

As a general principle, we find several basic concepts of adaptivity. The most relevant dichotomy of terms in this context is that of micro and macro adaptive approaches. The literature on adaptive systems uses the term macro adaptivity to mean systems that are only adaptable on a coarse granularity, for example offering a small number of variations for the overall system. Data used for macro adaptation is often gathered before adaptation takes place, and is not improved while the system is running. Micro-adaptive approaches on the other hand are characterized as systems that operate on a much more detailed level, adapting elements directly related to the current actions of a user. The data used for micro adaptivity is therefore often

3 [http://www.braingame-shop.de/epages/62390116.sf/de\\_DE/?ObjectPath=/Shops/62390116/Products/HC10](http://www.braingame-shop.de/epages/62390116.sf/de_DE/?ObjectPath=/Shops/62390116/Products/HC10), last visited on January 24, 2013

4 [http://www.braingame-shop.de/epages/62390116.sf/de\\_DE/?ObjectPath=/Shops/62390116/Products/HC10](http://www.braingame-shop.de/epages/62390116.sf/de_DE/?ObjectPath=/Shops/62390116/Products/HC10), last visited on January 24, 2013

5 [http://www.braingame-shop.de/epages/62390116.sf/de\\_DE/?ObjectPath=/Shops/62390116/Products/HC10](http://www.braingame-shop.de/epages/62390116.sf/de_DE/?ObjectPath=/Shops/62390116/Products/HC10), last visited on January 24, 2013

based on a diagnosis of the user and a feedback loop created while the system is used [107].

A slightly different definition of micro and macro adaptivity is introduced in the context of the projects ELEKTRA and 80Days described below. There, micro adaptivity relates to adaptive interventions changing aspects of the game inside of individual situations, such as providing a tip or adding an encouraging line in the dialogue with a character. Macro-adaptivity, on the other hand, refers to changes on the level of the game situations, choosing different variants of situations or different paths through the game. This unconfined scope makes macro adaptive interventions well suited for assisting players over the course of the whole game. In the case of learning objectives, the player can be lead towards an optimal set of objectives that are in line with his or her learning pace and previous knowledge.

However, macro adaptive choices can only be carried out between adaptive game units, and are therefore not as immediate as micro-adaptive interventions. Therefore, micro-adaptive interventions are well suited for immediate feedback or adaptation. The combination of both approaches allows the creation of applications that are highly adaptive to players both on the local and global scale.

### 2.2.1 Adaptive E-Learning

In educational settings, adaptivity has been used in many commercial and academic projects; see the account by Knutov et al. [79] for an overview. Adaptivity adds fundamentally to the experience of a learner since it allows the system to be aware of preferences and skills of the learner, leading to a personalized experience. By adapting elements of e-Learning systems, positive results have been shown, for example by Conlan and Wade [31].

Two areas where this approach has long been researched are Adaptive Hypermedia (AH) and Intelligent Tutoring Systems ITS. Adaptive hypermedia systems can use adaptivity mainly in the context of navigation, offering personalized navigation which leads users along an optimal learning path through the available educational content [133, 18] [138, chapter 8].

The realization of adaptivity is carried out using several models resulting from the application domain e-Learning. Shute and Towle [131] name the following models:

- A *content model* maps the learning domain and interdependencies between elements of a curriculum.
- A *learner model* (also referred to as “student model” in other sources) summarizes characteristics of the learner such as skills or learning preferences.
- An *instruction model* binding the two previously mentioned by assuring the learner is provided with the right information or assessment at the right time.

In order to carry out adaptation, an adaptation engine is realized as a component which handles the actual adaptation and the management of the involved models. For example, it can select the next content to present to the user or provide adaptive navigation of the content. Updates to the models are carried out by means of test-based assessment, involving closed exercises formats that can be machine-readable, including clozes, multiple choice questions or drag & drop exercises.

A common concept for creating e-Learning applications are learning objects, which are formalized in several standards as basic units of learning contents that can be combined to form educational courses. They have been researched for a long time in e-Learning, for instance, with a focus on data storage as described by Hörmann et al. [57]

Several standards for e-Learning exist. The SCORM (Sharable Content Object Reference Model) [1] standard is focused on the overall structure of educational courses and the organization and sequencing of learning objects in the part “Sequencing and Navigation” (available since version SCORM 2004). The LOM (Learning Object Metadata) [61] standard defines a set of metadata for the description, retrieval and use of learning objects.

### 2.2.2 Adaptive Storytelling

Adaptive storytelling is a research direction which unifies narrative-driven game-like experiences and adaptive control of stories. Due to this combination, it is relevant to examine the existing work on non-linear, interactive storytelling.

In an interactive drama, the player is given control of a character in the narrative and can therefore influence it actively. The storytelling system meanwhile monitors all actions of the player, including the communication of agents in the world, and adapts the narrative and the actions of players based on this input. The interactive control of the story and the adaptation based on the actions of the player is commonly realized centrally in the form of a “drama manager”, a component responsible for keeping up the narrative flow and regulating the tension of the story, see for example [91]. Similarly, character-driven storytelling systems such as [24] use a comparable architectural pattern where high-level actions of simulated characters cause actual changes in the game as visualized by the game engine.

Noteworthy experiments in interactive storytelling are those by Chris Crawford (Erasmatron<sup>6</sup> and Storytron Story Engine<sup>7</sup>) or by Andrew Stern and Michael Mateas (Façade) [97]. While these experiments were able to demonstrate the approaches with impressive results, the field of interactive storytelling has not yet seen commercial success.

The work on interactive storytelling can be structured as the study of models for narrative structures on the one hand and the interactive control of the storytelling process on the other hand.

One of the largest problems arising due from the adaptive nature of interactive storytelling is the conflict between the interests of an author of interactive storytelling and those of the person controlling the interactive storytelling system. Whereas an author intends to plan the narrative to create a suspenseful and appealing story and therefore wishes to control key plot elements, players wish to make choices freely without narrative constraints. Therefore, an author has to set up the rules for this control, later yielding the final decisions to a player. This conflict is referred to as the “narrative paradox”, which Louchart and Aylett [88] researched based on experiences with pen and paper role-playing games, where a game master controls the narrative. The game master, taking a similar role as the drama manager, is in charge of adapting

<sup>6</sup> <http://www.erasmatazz.com/>, last visited on January 24, 2013

<sup>7</sup> <http://www.storytron.com/>, last visited on January 24, 2013



| NAME               | FIELD                  | ASSESSMENT  | INTERVENTION                                     |
|--------------------|------------------------|---|--|
| Madame Bovary [25] | Interactive Narratives | Speech recognition and motion data interpreted as actions               | Planning-based                                   |
| Façade [97]        | Interactive Narratives | Interpretations of player utterances and actions                        | Macro-Adaptive: Choice of appropriate story beat |
| Scenejo [137]      | Interactive Narratives | Dialogues with players  | Emergent character-based                         |
| FAtiMA! [82]       | Interactive Narratives | Interactions with players based on previous mixed-initiative rehearsals | Emergent character-based                         |

Table 1: Overview of cited examples for adaptive interactive storytelling systems.

an abstract design for the game’s narrative by reacting to the actions of players and keeping in mind the suspense arc and quality of the narrative.

In a similar way, story pacing as described by Göbel et al. [47], i.e. the “rhythm” of a narrative or the timing and sequence of narrative events, is harder to realize in digital media than in other media used for storytelling, and the player’s perception due to the effects of non-linearity and interactivity, can differ from what the author intended. As Crawford explains [32, p. 34], interactive storytelling systems can offer interactivity, variety and story pacing simultaneously, at the cost of reduced quality of the narrative itself compared to a linear, non-interactive narrative.

Apart from systems that are based on narratology, other common systems use artificial intelligence techniques in order to build interactive storytelling. For example, the interactive narrative of “Madame Bovary” [25] uses planning algorithms in order to allow virtual characters to have individual goals and attempt to fulfill them in the course of the storytelling application. The resulting conflicts between interests of virtual characters and the player create a suspenseful story at runtime. Many current digital storytelling authoring systems make use of this emergent narrative approach, based on virtual agents. These include the Scenejo system [137], which is mainly focused on configuring conversational agents, approaches using rehearsals for demonstrating the envisioned story to virtual characters [82, 5], or authoring during the playing experience [146].

### 2.2.3 Adaptive Educational Games

Adaptivity is directly applicable to games, and games can benefit from the effects of adaptivity. By virtue of the same arguments for educational games in general, adap-



tivity in games should not focus either the learning or playing aspects, but should optimize the balance of the two. While the learning effectiveness is an important factor, the motivation and interest of players should be increased by adaptation.

For example, the feeling of flow, where players are challenged but not overstrained, can be upheld by using adaptive methods. Adaptive elements in games can be the difficulty, the behavior of non-player characters, adaptations of the game's narrative or the game rules. However, for a successful adaptive game, the adaptivity should not be a visible feature of the game but instead be minimally disruptive. The concept of observing a player's behavior passively and interpreting success or failure as game evidence [109] is preferable to interrupting play using methods like a questionnaire to obtain information for adaptation.

Adaptivity for games, is realized by establishing a "player model" subsuming aspects and characteristics of players and using this model for adaptation purposes.

In commercial games, adaptivity has mainly been used in the dynamic adaptation of difficulty. Systems as those described by Hunicke [60] evaluate the performance of players and choose an appropriate difficulty. The game "Left 4 Dead" features a "drama manager" (similar in spirit to those of adaptive storytelling) that analyzes the state of the game and alters game situations when appropriate to create tension and exciting experiences [15].

Maciuszek and Martens [89] describe an effort to combine the strengths of role-playing games with intelligent tutoring systems, leading to an architecture for adaptive educational games. A central drama manager uses adaptive algorithms to align game elements such as characters or items with the educational goal of the game.

An initial approach to player modeling is introduced by Houlette [59]. In this work, a generic and extensible player modeling framework is presented. While this framework was applied to the adaptive control of non-player characters, it is also applicable in the context of story-driven games.

Thue et al. [143] describe PaSSAGE (Player-Specific Stories via Automatically Generated Events) [144, 145], a an approach towards adaptive storytelling based on player modeling. Additionally, a test game is presented and the effect of adaptivity evaluated empirically.

Another player modeling approach is described and evaluated by Sharma et al. [129]. The "Anchorhead" method allows players to rate plot points (events in the narrative). When the narrative has finished a "player trace" is created from the ratings. A database of player traces is used during subsequent playthroughs in order to compare players (based on their actions and decisions) with it and to choose the most appropriate plot points for them. This approach therefore requires prior playing of the game in order to build the repository of player traces and cannot be used immediately.

Charles et al. [27] choose a very similar approach in which a set of player types is required which are used to assess the actions of the player. The approach is not targeted at digital educational games and there is no implementation.

Magerko et al. [92] analyze the possibilities of adaptive educational games with a focus of adapting game mechanics, input options and graphical user interface. The model for the player is not assembled online (i.e. during play) and has therefore be created a priori.

Rowe et al. [124] describe the storytelling-based game adaptive educational game "Crystal Island" that uses an adaptive dual planning-approach in order to create a

narrative adapted to the player. The game uses a separate knowledge space and narrative space and corresponding components (tutorial and narration planner) that can highlight a “goal” in the game. The game then attempts to work towards a common goal. The planners are based on a planning algorithm called “Shop2” and use the NPCs of the game in order to teach using the narrative and adaptive dialogues. They analyze the actions of players and react to violations of the goals by finding new goals or adding actions to the knowledge or narrative space in order to reach to original goal again. For this purpose, the algorithm requires a set of data that models a priori knowledge of play and learning processes in the game in order to assess the goals of players. The game is realized as an open virtual 3D world.

Burgos et al. [21] describe a prototypical implementation of an educational game authored using the e-Adventure authoring tool for point & click-based educational games. Adaptivity in e-Adventure is mainly realized as different difficulty levels. The capabilities of players are assessed in a questionnaire before play begins and difficulty levels for individual sections realized using this method.

The project ELEKTRA [71, 75] provides an adaptive learning environment in which players can move around in a 3D environment filled with tasks from physics and engineering involving magnetism and light. The narrative of ELEKTRA is linear and therefore not adaptable, and does not play a large role in the game. Adaptation is carried out by interpreting actions of the player (such as “lights a torch”) to infer the knowledge state of the player. This state is represented as a set of probabilities, specifying the probability that a certain skill has been mastered by a player. These skills are modeled using Competency-Based Knowledge Space Theory (CbKST) [55], which is shown in detail in chapter 5 as the basis for adaptivity concerning learning in our concept. Adaptation is carried out solely on the educational content, in the form of supporting interventions (help texts, NPC dialogues, changes in difficulty, among others). These adaptations are drawn from a pool of cognitive, meta-cognitive or motivational hints that are selected based on the most important need of the player. In the terminology of the ELEKTRA project, this adaptive mechanism is referred to as micro-adaptive.

The successor project to ELEKTRA, 80Days [73, 70], focused research on a game architecture allowing the adaptation of an educational game not only using local changes but also using global changes (choosing adaptively between learning paths). In this game, which is used to create geography, macro-adaptive choices have also been allowed in order to adjust towards learning objectives during play. Since the research in this thesis has been partly carried out in the context of the 80Days project, elements of the concept and implementation are influenced by work on 80Days. For a detailed description of 80Days as an example for games that can be authored using the authoring tool StoryTec, see Appendix A.2.

A novel way to adapt games is the application of procedural content generation methods, which create game content (for example, the layout of a level or textures applied to objects) at runtime without human intervention, governed by a set of pre-configured parameters [86]. Smith et al. [134] present a system in which new content for upcoming sections of a game is created based on the current state of the game.

| NAME                  | FIELD         | ASSESSMENT                                       | INTERVENTION   |
|-----------------------|---------------|--|--|
| Hamlet [60]           | Entertainment | Probability of death                             | Micro-Adaptive: Provide items, adjust hit points, etc.   |
| Left 4 Dead [15]      | Entertainment | Estimation of group emotional state              | Micro-Adaptive: Regulate numbers and timing of enemy encounters                                      |
| RPG-based ITS [89]    | Education     | Initial Assessment, Updates                      | Macro-Adaptive: Choice of appropriate quests from Knowledge Space                                    |
| (Houlette) [59]       | Entertainment | Player action interpretation, Update via LMS     | Adaptation of AI behaviour   |
| PaSSAGE [144]         | Entertainment | Annotations of player actions                    | Macro-Adaptive: Choice of appropriate next event   |
| Anchorhead [129]      | Games         | Feedback by players on each plot point           | Macro-Adaptive: Choice of appropriate next plot points   |
| (Charles et al.) [27] | Games         | Monitoring of player performance (not specified) | Not specified  |
| S.C.R.U.B. [92]       | Education     | A priori: Questionnaire                          | Macro Adaptive: Different game configurations  |
| Crystal Island [124]  | Education     | Interpretation of student activities             | Micro-Adaptive, based on planning  |
| e-Adventure [21]      | Education     | Questionnaire, A priori                          | Macro adaptive: Choice of different difficulty levels  |
| ELEKTRA [70]          | Education     | Player activity interpretation                   | Micro-adaptive: Hints, character dialogues   |
| 8oDays [73]           | Education     | Questionnaire, player activity interpretation    | Micro-adaptive: Hints, character dialogues; Macro-Adaptive: Choice of appropriate learning situation |

Table 2: Overview of cited example adaptive game systems.

## 2.3 AUTHORIZING TOOLS

After the presentation of basic technologies for adaptive educational games in the previous sections, we now examine the related work in the field of authoring tools. The authoring tools in this section are sorted into the categories multimedia, e-Learning, interactive storytelling and educational games, roughly following the sequence of the establishment of authoring tools in the respective area.

The concept of providing an authoring tool that streamlines the workflow in a certain area has a long history in computing. It is successful due to several factors. First, it allows non-technical users to work on projects that would otherwise be out of their reach (due to lack of expertise, especially concerning programming languages). It can bring structure into unstructured domains (such as game development) and it can speed up development by streamlining and automating common tasks. Authoring tools also reduce coordination effort between different author groups involved in a production process such as writers and artists.

Multimedia authoring tools have been developed after groundbreaking developments in multimedia and the steadily increasing computing power especially of consumer electronics shifted the focus of multimedia production from algorithmic problems to content production. Initial multimedia systems were focused on the work of a programmer, while later on systems shifted to frameworks into which multimedia content could be entered into. Authoring tools here assist content producers in integrating the content into multimedia products. A general overview of practices in multimedia authoring tools is given by Bulterman and Hardman [20], who provides the following list of paradigms that underlie multimedia authoring tools:

- *Structure-based*. Based on an abstract model of the authored application, objects are placed in the structures defined by these models.
- *Timeline-based*. Objects and actions are placed on a timeline, indicating the temporal order of events.
- *Graph-based*. Actions in the authored system are configured using a visual, graph-based approach.
- *Script-based*. Authoring tools based on this paradigm expose a programming language to users, which opens the system to be programmed but simultaneously introduces the need for understanding programming languages.

General-purpose multimedia authoring tools have reached commercial success and are widely used, especially in the context of web-based multimedia. Authoring tools here include products for web page development such as Adobe Dreamweaver<sup>8</sup> and those for the creation of interactive multimedia applications such as Adobe Flash<sup>9</sup>. In the following, we focus further on authoring tools with dedicated features for the specialized fields e-Learning, interactive storytelling and educational games. This additional focus is necessary since multimedia authoring tools are very general and do not support the task of creating adaptive educational games specifically.

<sup>8</sup> <http://www.adobe.com/products/dreamweaver.html>, last visited on January 24, 2013

<sup>9</sup> <http://www.adobe.com/products/flash.html>, last visited on January 24, 2013

### 2.3.1 *E-Learning Authoring Tools*

Authoring tools for e-Learning are in use at many public institutions such as universities as well as in private educational institutes or for in-house corporate training. In this area, experts can enter content for learning scenarios without the need to be trained in programming or web site creation. The resulting products (educational courses, for example offered via the internet as a web-based training) are courses which are usually assembled from a set of learning objects.

The Resource Center[57] authoring tool features web-based authoring and can be used to create SCORM-compliant courses. It was later renamed to “docendo” and published as open source software by the multimedia communications lab (KOM). A commercial example which also follows the visual programming paradigm is Macromedia Authorware<sup>10</sup>. A comprehensive overview is given by Horton and Horton [58].

Tools in adaptive e-learning are often based on existing e-learning authoring tools which allow the creation of learning objects and add the possibility to control adaptive features. An example of this is described by Bontchev and Vassileva [14].

Generalizing over the large set of authoring tools for e-learning, we find three main characteristic features many of the authoring tools share:

- *Support for setting up navigational structures.* Since many e-Learning products are realized as a set of interconnected hypertext pages, the simplified setup of navigational structures is central to e-Learning authoring tools.
- *Encapsulation of technical details.* Instead of having to work on a low technical level, e.g. on the source code of HTML files, authors work on a higher level of abstraction while the authoring tool carries out technical tasks automatically.
- *Support for templates.* Interactive elements in e-Learning courses (such as drag & drop tasks, quizzes or closes) are presented to authors often as templates, i.e. structures which allow content (images or text) to be entered into fields, which are then realized as interactive elements.

### 2.3.2 *Interactive Storytelling Authoring Tools*

Since Interactive Storytelling is a domain driven by creativity, the creation of authoring tools has been a focus of research in this area. The Network of Excellence IRIS (Integrating Research in Interactive Storytelling) [26] provides a study in which several authoring tool of interactive storytelling were compared. Spierling and Szilas [136] bring forward the following two recommendations based on this study which simultaneously illustrate the major challenges faced by interactive storytelling authoring tools:

- *“Listen to authors:* Make tools that better match the concepts and practices of media designers and content creators
- *Educate potential authors:* Make procedural principles of Interactive Storytelling understandable”

<sup>10</sup> <http://www.adobe.com/products/authorware/>, last visited on January 24, 2013

Both of these recommendations are also relevant to educational game authoring tools and are used in the definition of requirements on our authoring tool concept.

The focus of authoring tools for interactive storytelling is on creating interactive experiences which are based on dramaturgic structures. One line of research can be found in the authoring tools Cyranus [64] (used in the project art-E-fact), U-CREATE [48] and INSCAPE [34]. The user interface of INSCAPE is shown in figure 2.

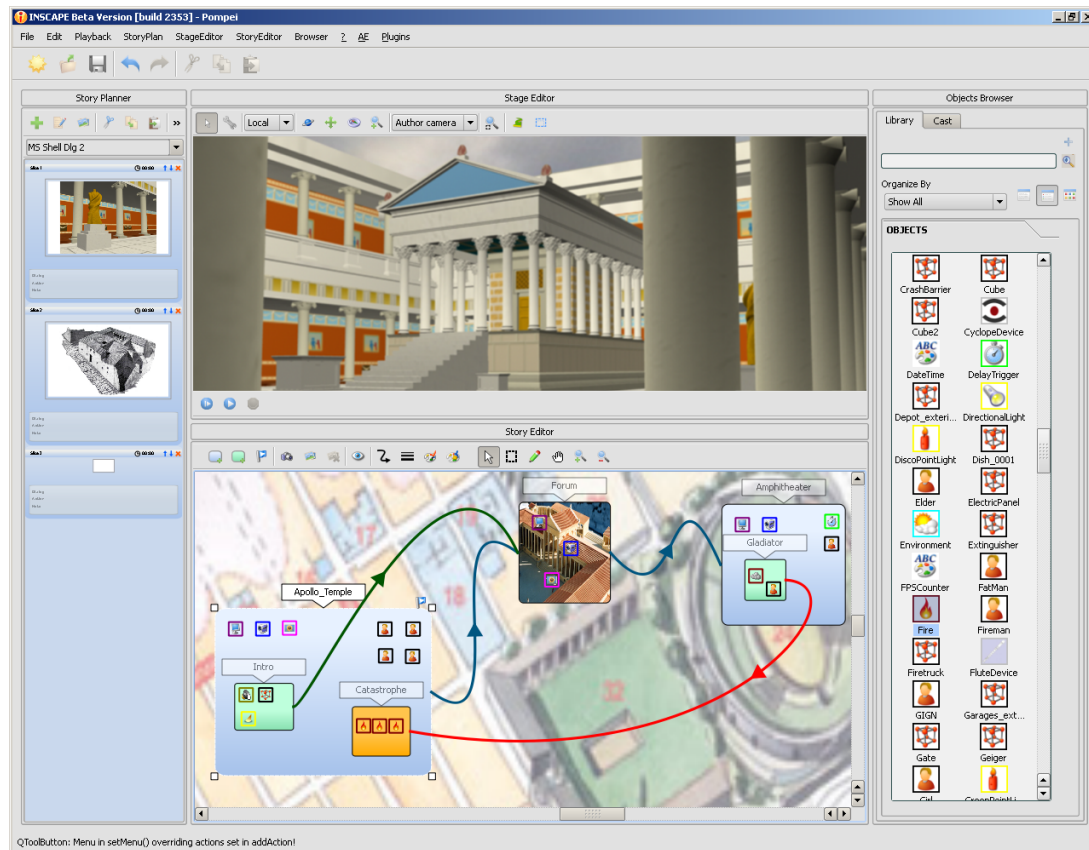


Figure 2: The user interface of INSCAPE.

The user interface of INSCAPE is sub-divided into four main elements. On the left, we find the “Story Planner”, a component used to create storyboards for the authored narrative. The center of the user interface is composed of the “Stage Editor” (top) and “Story Editor” (bottom). The former shows the currently selected part of the narrative in a 2D/3D view where objects can be manipulated. The latter visualizes the structure of the whole game and allows selection and editing. On the right, the “Objects Browser” can be found, which provides a collection of available objects to be used in the narrative.

Narratives in INSCAPE are modeled in a model with three layers. The topmost level of “Story” encompasses the whole narrative. On the next level, “Stages” can be found that separate the narrative spatially, similar to levels in computer games. The lowest level, “Situations”, is concerned with the actual narrative and the actions of the player and of characters. This model is saved and exchanged using the format INSCAPE Communication Markup Language (ICML).

Interaction is realized in INSCAPE mainly by the concept of events that are associated with game events. An example is a trigger area that triggers an event when the



user's avatar enters it. The actual reaction of the system to the event is programmed by the author using the LUA scripting language, supported by a library of LUA snippets.

Adaptivity is included in INSCAPE by the mechanism of "Story Pacing" [47]. Hereby, authors can create "Strategies" which are in effect adaptive changes to the game such as a "limit time" strategy that introduces countdowns to the narrative. A set of strategies is activated by choosing among a set of "Story Policies" such as a beginner mode that features hints and no time limit. There is no dedicated mechanism for assessment or user modeling.

The Scenejo [137] authoring system is an example for an authoring tool supporting the emergent character-based approach of storytelling introduced in section 2.2.2. Here, authors model the narrative application mainly by configuring dialogues, which is realized on several layers. On the lowest layer, "stimulus-response elements" are configured as parts of a character's dialogue using the language Artificial Intelligence Markup Language (AIML). On the layer above, these dialogues are visually connected to form a dialogue graph.

### 2.3.3 Educational Game Authoring Tools

Examining the history of games (e.g. Montfort and Bogost [98], who highlight the production processes of games in the Atari 2600 era, beginning in 1977), it can be seen that initially, games were limited severely by the available computing power. Early arcade games or home console games had to suffice with hardware that could only display tens of simple vector or pixel objects. The major challenge in this phase of game development stemmed from optimizing games and employing the correct algorithms to make games run at a minimum frame rate. Early games were often developed by single programmers who worked simultaneously on game design, primitive artwork and the whole programming.

Over the years, game hardware became more powerful and games added more multimedia features. Concurrently, game teams increased in size and tasks were separated and given to specialists. In order to create games, tool "pipelines" were created which define the tools and processes (including naming and other conventions) that result in content that can be added to games. In order to assemble games, most game developers today choose a data-driven approach in which game content is not directly embedded in program code but instead management components are responsible for setting up the game world before play. Complete software frameworks for the creation of games are referred to as "game engines". These management components are often supplied with input files which are created in game editors. These editors most often are developed in-house and therefore are very specialized and not easy to use for untrained users. Their use is complex and requires knowledge about basic concepts and development paradigms as well as the knowledge of programming languages.

For more general purposes, there exist a multitude of game creation tools with various degrees of complexity. This game middleware is created by companies in order to license to other developers. One commercial example is the Unity3D<sup>11</sup> game creation software, which features a strong continuity between the authoring phase

<sup>11</sup> <http://unity3d.com/>, last visited on January 24, 2013

and the final game by offering a What-you-see-is-what-you-get (WYSIWYG) view of the game during authoring. Another example which was introduced as a tool for teaching programming is the GameMaker system [105].

Several general-purpose authoring tools or game editors for adventure games are available. The free Adventure Game Studio (AGS)<sup>12</sup> has generated a large community of independent adventure game developers. The Visionaire Studio<sup>13</sup> was used in the development of the educational adventure game Winterfest [93].

Authoring tools for educational games are those game authoring tools which have been created with a focus on education, supporting in some way the creation of such games and accounting for the specific differences to entertainment game development. Tools can be aligned along a spectrum ranging from very complete tools that allow the creation of arbitrary games in any genre on the one hand to tools that are confined to one genre or one specific game only. However, while the expressiveness of authoring tools decreases along this spectrum, we find that the usability and ease of use increases since the specialized tools can support users far better as the tool providers know in advance which kind of game authors will create. Figure 3<sup>14</sup> illustrates this spectrum with very powerful yet also challenging tools on one side and more restricted but also easier to learn, specialized tools on the other side.

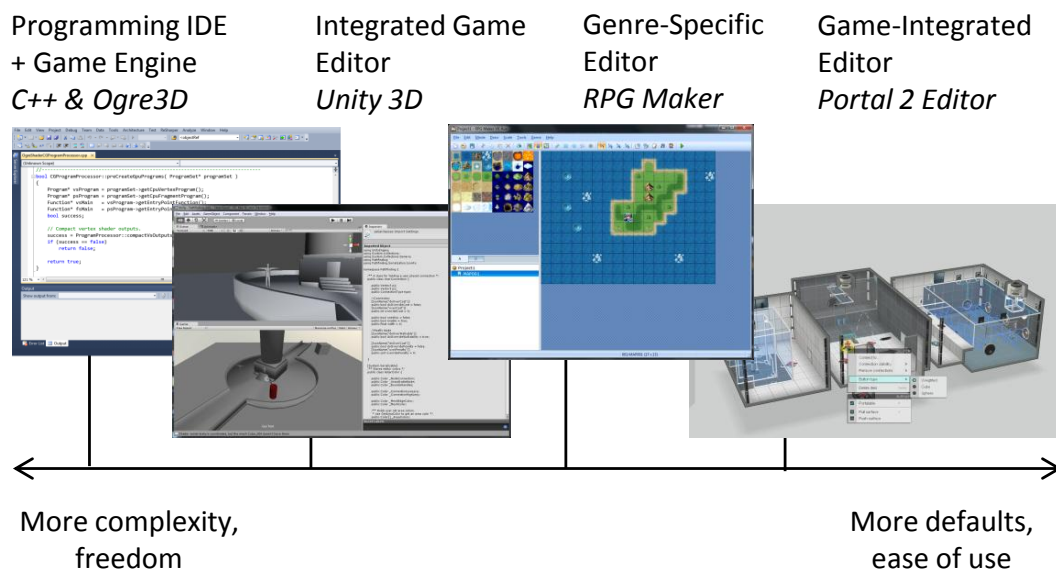


Figure 3: The spectrum of authoring tools and game editors.

Comparing the authoring tools in e-Learning, multimedia and games from the previous sections, we find that the inherent complexity of the authored artifacts (from e-Learning courses to educational games) increases. A typical e-Learning course often consists only of navigational elements to move from one learning object to another, or simple interactive elements such as multiple choice questions or clozes. Assessment and feedback to the learner is typically not carried out while such interactive

<sup>12</sup> <http://www.adventuregamestudio.co.uk>, last visited on January 24, 2013

<sup>13</sup> <http://www.visionaire-studio.net/cms/adventure-game-engine.html>, last visited on January 24, 2013

<sup>14</sup> Examples used: Ogre3D (<http://www.ogre3d.org/>), Unity3D (<http://www.unity3d.com>), RPG Maker (<http://www.rpgmakerweb.com/>), Portal 2 Editor (<http://steamcommunity.com/sharedfiles/browse/?appid=620>), all last visited January 24, 2013



elements are being used, but instead after the user has finished a task. Games, on the other hand, are often real-time simulations which process player input constantly, and players expect immediate feedback to their actions.

While e-Learning tools could be used to build simple educational games using the predominant exercise types, this would limit the authoring to this narrow subset of possible games and simultaneously would exclude the use of game mechanics that differentiate games from typical e-Learning products.

In the area of educational games, we find a small number of authoring tools with specialized features for the serious purpose of the game. The e-Adventure authoring tool [99, 148, 149] provides, apart from the functionality of creating adventure games, the possibility to export games to Learning Management Systems [35]. Another feature aligned to the purpose of education is the creation of books that can be read in game to transport knowledge textually.

The SHAI Scenario Editor described by Est et al. [41] allows users to define educational scenarios using high-level logic visualized in diagrams that is converted to low-level logic internally by the system.

Table 3 provides an overview of the authoring tools analysed here, along with a categorization into the authoring paradigms by Bulterman and Hardman [20].

## 2.4 GAME MODELS

As a preparation for providing a model for adaptive educational games, we present the possible ways in which structure and content of interactive, adaptive applications and games can be modeled and the choice of a model to be used subsequently in the authoring tool concept is motivated.

Lindley [84] (see also figure 4) described the differences between games that are governed by storytelling, pure play or simulation or a mixture of these aspects. It can be noted that games that are governed by storytelling often exhibit the most structure, since a narrative is often naturally broken down into acts and scenes. On the other hand, simulations and pure games can be seen as software which exhibits behavior that is mostly governed by the simulation of reality or the simulation of game rules. Examples are games with no narrative such as Tetris or strategy games. While models for such gameplay are also possible, they usually have to expose the game rules as part of the model. These models cannot be subdivided as easily and distributed over several scenes. This would lead to an authoring process that is less structured than the one described in this thesis. Therefore, our focus lies on authoring of storytelling-based educational games or those which can naturally be decomposed into interconnected sets of scenes.

For modeling multimedia applications in general, we find the Multimedia Modeling Language (MML) [111] which uses methods of model-driven software development to derive a model for interactive applications. An application is modeled on the lowest level by defining application entities along with executable methods and parameters. On the next level, the application is modeled in “presentation units” referred to as scenes, which are connected in the fashion of a finite state machine. Interactivity is modeled by extending UML activity diagrams, connecting user interface elements, application entities and their events. The result is translated into a code skeleton that can then be used to build the actual application in an author-

| NAME                  | FIELD                       | PARADIGM         | NOTES                                  |
|-----------------------|-----------------------------|------------------|--|
| Resource Center [57]  | e-Learning                  | Structure        | Open-source version docendo available  |
| Cyranus [64]          | Interactive Storytelling    | Graph            | Results used in INSCAPE                |
| U-CREATE [48]         | Interactive Storytelling    | Graph            | Results used in INSCAPE                |
| INSCAPE [34]          | Interactive Storytelling    | Graph            | Story Editor as novel component        |
| GameMaker [105]       | Games                       | Structure/Script | Intended for teaching programming      |
| Adventure Game Studio | Adventure Games             | Structure/Script | Large user community                   |
| Visionaire Studio     | Adventure Games             | Structure/Script | Used for educational games             |
| e-Adventure [149]     | Educational Adventure Games | Graph/Structure  | Support for manual adaptation          |
| SHAI [41]             | Educational Games           | Graph            | High-level logic diagrams as interface |

Table 3: Overview of cited authoring tools.

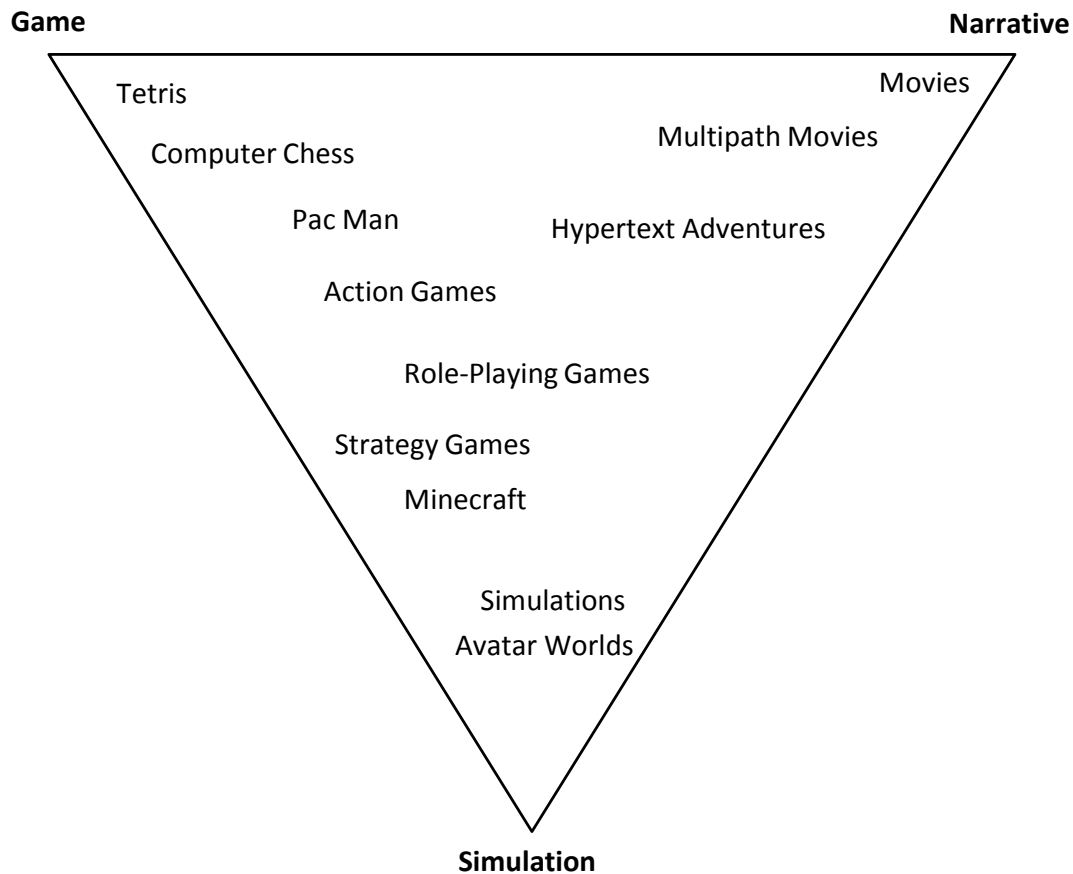


Figure 4: “Ludic Space” as described by Lindley [84] (adapted from [84]).

ing tool (demonstrated using Adobe Flash). However, no direct manipulation of the model in the authoring tool is aimed for.

A similar language, albeit with an explicit focus on games, is the EGGG Language (Extensible Graphical Game Generator) described by Orwant [104]. Here, games are defined by their rules, which are transcribed in a natural-language programming language. A game is automatically generated, including the necessary graphics. Due to the focus on rules, games that can be modeled with this language are limited to the “game” side of Lindley’s ludic space [85].

#### 2.4.1 E-Learning Models

Examining authoring tools from e-Learning, we find that the major authoring process lies in the creation of learning objects and the creation of navigational structures which show how they are connected to each other.

Learning Objects (LO) or e-Learning Objects (ELO) were named in the context of web-based learning. Learning objects are defined by the IEEE as: “Any entity, digital or non-digital, which can be used, re-used and referenced during technology-supported learning” [61]. Polsani [112] defines: “A Learning Object is an independent and self-standing unit of learning content that is predisposed to reuse in multiple instructional contexts”. However, no single definition has become universally accepted.

The essences of the available definitions are the aspects “independence” and “re-usability”. Learning objects are intended as self-contained learning units or chunks, which can be assembled in larger contexts (e.g. as a web-based learning course). For categorization, management and distribution to users, metadata for learning objects and their relationships have been developed. Major standards for metadata are LOM [61] and the Dublin Core Metadata Initiative (DCMI) [62], the latter especially for relationships definitions between learning objects (e.g. “isRequiredBy”).

In typical e-Learning tools, users can create individual learning objects for example as web pages. These objects are then inserted into larger structures, for example in the form of chapters that group related learning objects in an order that follows the optimal learning path of the curriculum. This is comparable to a tree that users usually traverse in a depth-first way.

An alternative are adaptive hypermedia e-Learning tools, in which learning objects are chosen or suggested based on the suitability for a continuously updated learner model.

#### 2.4.2 *Interactive Storytelling Models*

The next area we turn towards is interactive storytelling, which offers several different models for interactive stories, which are conceptually very close to adaptive games. The main models examined here are those based on the concept of scene as compared to those inspired by artificial intelligence approaches to interactive storytelling. We subdivide the models found in this area into those we refer to as scene-based and those that are based on artificial intelligence, e.g. agent-based models.

The first of these models is inspired by the use of “scenes” as found in narratives in theater, novels and movies. Scenes are understood to represent situations of the story, and in their sum result in the overall story. Scenes can be represented as nodes of a story graph, with directed edges indicating that there is a way to reach the successor scene from the predecessor scene. Such a transition is usually triggered by an action of a user. The application ends when its state has advanced to the final, end scene as described by Sheldon [130].

Four types of graphs can be found in practice:

- Strictly linear stories - directed, acyclical with no node adjacent to more than one edge, see figure 5a.
- A branching story with possibly different endings - series-parallel digraphs, see figure 5b.
- A main story that offers optional side tasks or “quests” - union of a linear story graph and several sub-graphs for the quests which are available as branches from the main story line, see figure 5c.
- A set of nodes that are chosen adaptively based on the current story state - “scene cloud”, fully connected graph with (implicit) edges not drawn for clarity purposes, see figure 5d.

All of these graph forms entail advantages and disadvantages concerning complexity of authoring, the amount of control over the story (by the author and the player, respectively) and the flexibility and adaptability provided.

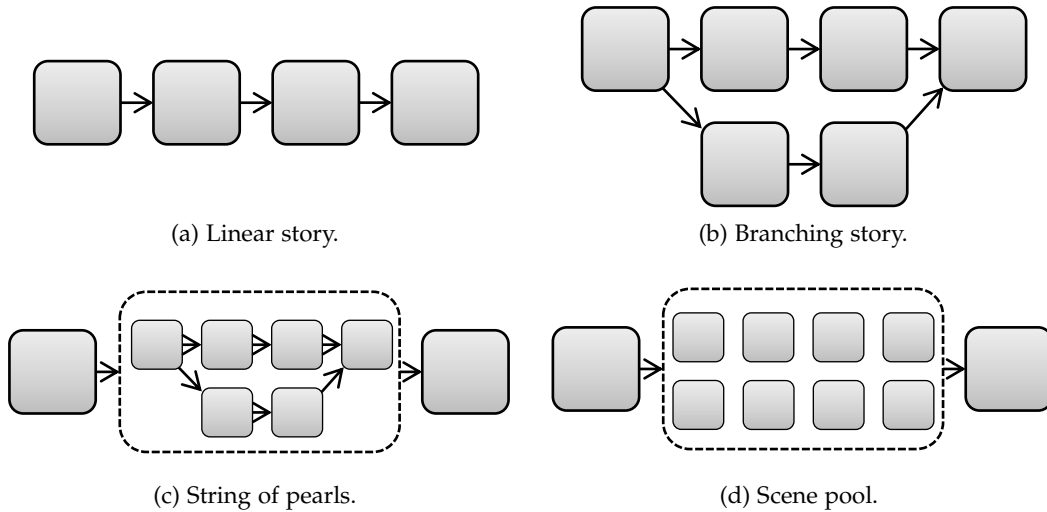


Figure 5: Story Graphs.

The linear story is very similar to the writing process of a book or a screen play, and therefore the least complex and intensive for a story author. Since the plot is traversed in a linear fashion, the author has complete control over the order of scenes and the narrative context that the player has. However, as there is no possibility for branching, adaptivity and player choices affecting storytelling are not possible in this story graph form. Many commercial games are structured in this way.

The linearity of the first model is overcome by allowing branches in the second model. The player is given choices, which lead to different branches of the story and possibly also different ending states. This format has the possibility of creating (the perception of) influence on the story on the side of players, and encourages replaying the game since other choices open new story branches. For an author and the overall production of the game, this form entails more work, since additional content has to be created. The additional work can be limited by realizing the story not as a true tree but rather as a foldback scheme<sup>15</sup> (cf. [32, p.126f]), resulting in series-parallel digraphs. However, this model furthermore requires authors to keep in mind that players can have information about the narrative from different branches, which has to be synchronized at important plot points. This is one reason why many games use a “string of pearls” model, in which branches are always lead back together at important story points in order to make sure players have the same knowledge.

Similarly, a linear story with optional side quests solves this problem by enforcing the linear structure of major, important story events, while relatively unimportant and self-contained side narratives are offered freely to the player.

The final model shifts the control over the story almost completely from the author to the player, by making the story continuation dependent upon player choices and the adaptive choices of the system made based on the player’s models. Authors have to be aware of the underlying adaption mechanics and adapt their storytelling to them. If the individual scenes are not intended to be completely self-contained but be parts of an overall story, the authors have to specify pre-conditions for scenes to

<sup>15</sup> A structure in which different branches of a story lead back to a shared scene, which nivellates the previous choices of players.

Listing 1: An example logical description of a scene in “Madame Bovary”. (Source: [25])

---

*( Action : Emma-Asks-Rodolphe-To-Run-Away-Together*

Preconditions :

*affinity (Emma, Rodolphe , HIGH)*  
*power-over (Rodolphe , Emme, HIGH)*  
*affinity (Emma, Charles , LOW)*  
*anger-towards (Emma, Charles , HIGH)*  
*accepts-adultery-risk (Emma, Charles ,*  
*HIGH)*

Effects :

*asked-escape-with (Emma, Rodolphe)*  
*womanhood (Emma, HIGH)*  
*excitement (Emma, HIGH))*

---

impose a loose order of the scenes in the scene pool. This approach serves as the basis for emergent narrative storytelling systems [42, 88].

A possible implementation of a scene pool can be found in Façade [97], which uses “beats”, capturing atomic dramaturgic events as a sequence of audio-visual elements. A beat covers a small part of the game’s story, and can be interrupted by another beat. If a beat is interrupted, it can be continued later on if it still fits logically and from a narrative perspective. New Beats are chosen from a Bag of Beats and then executed. Furthermore, a Drama Manager manages the dramaturgy of the story by choosing beats by following the structure of a suspense curve (raising tension until the climax, declining tension afterwards).

Opposed to scene-based approaches, other interactive storytelling systems model an interactive narrative as the result of the interaction of virtual characters and the player. For example, the “Madame Bovary” system [25] describes situations in the narrative using a logic-inspired domain-specific language in which the preconditions and effects of scenes are stated in terms of propositions about characters and their relationships. The simulation of the characters results in drama since characters can have conflicting motivations and goals. Listing 1 shows an example script in which a scene is described in which two characters decide to run away together.

The resulting authoring process is therefore conceptually further apart from the practice of writing a linear, non-interactive story than scene-based approaches. Instead of thinking about scenes and the relations among them, authors have to focus on the properties of characters and are required to have an understanding of the simulation that controls the narrative during runtime in order to gauge how the narrative might evolve during runtime. Since this process is error-prone, authors might have to engage in bug-fixing similar to regular programming languages since changes might have unexpected consequences in the game.

### 2.4.3 *Game Models*

The development of digital games is a very unstructured and diverse domain, and has given rise to far fewer industry standards for programming and content management than other fields. While business applications commonly rely on standard architectures and are developed according to software engineering best practices, the development of games lacks clear best practices. This is due to a range of reasons. First, there is a need to develop cutting-edge games and game engines, as digital games are often seen as benchmarks of current hardware platforms and users expect each generation of game to surpass the previous. Furthermore, the lack of a common syllabus or degree program shared between game developers, the creative nature of games and the strict deadlines and time-to-market schedules are factors in this. Therefore, no single common way of organizing game structures and content can be identified in the case of digital games. This section however represents the attempt to generalize some of the patterns commonly found in game programming and game data organization.

The basic unit with which games are usually structured is that of a level, a term used in a very broad sense. Commonly, a level is defined in a spatial way, indicating a part of the overall space in which the game is set. For example, in a 2D sidescrolling game, a level commonly stretches horizontally across several screens and has to be traversed by the player to finish it, while a level in a 3D game is usually a delimited 3D terrain in which the player has to accomplish certain tasks according to the game's genre. The concept of level is found in game editors such as Unity3D. From an author's perspective, the approach of using levels usually consists of the creation of the level architecture, in which objects are placed and controlled by attaching scripts to them. The way in which the player experiences the game is then mostly determined by the script code that is attached to objects in the level. The organization of game content in this way is, especially to a user not able to program, very hard to understand, since the behavior of game objects is controlled by using code.

An alternative to this way of organizing game content and logic is often found in games which feature a large, seamless world for the player to explore. This is often the case in computer role-playing games, in which players can move through a very large terrain. Game tasks are then organized into "quests", a set of tasks the player has to accomplish which is commonly linked to the story of the game, advancing the narrative upon completion or during the attempt of the quest. Game editors for such games can feature this concept also as a structure to be used by authors, where quests are created as a set of stages to be completed by a player.

Apart from this first axis on which game models can be examined concerning spatial organization, a second axis is spanned between data-driven and programmer-driven approaches. Early games tended to be programmer-driven since games had to be highly optimized and no toolchains for content production existed. For example, game graphics were not loaded from bitmap files but rather drawn directly from the program code, resulting in a high cohesion between the game programs and their content. Over the course of maturation of the game-industry, the approaches to game development have become more data-driven. Instead of integration of content and code, code is encapsulated in the form of an engine which loads data files including all content of the game. This data-driven approach also allows the creation of game

editors, in which non-programmers are able to create data files for games (such as level files or 3D assets) without having to manipulate the engine code.

An active area of development is the application of model-driven development to the creation process of games.

[122] present a process where game design and programming is approached using techniques from model-driven development.

The model underlying the e-Adventure authoring tool has been described as a visual domain-specific language which supplies authors with a visual editor for structuring the game's high-level structure by Marchiori et al. [95].

The system by BinSubaih and Maddock [12] separates a game's core logic from the actual implementation in a game engine. This is achieved by modeling the game in the so-called "G-Space" which can be executed on several game engines by providing adapters for these engines.

## 2.5 AUTHORIZING SUPPORT FOR NON-PROGRAMMERS

The interactivity and complexity of games necessitates a form of programming. Non-programmers can be supported by including natural language or visual programming languages. A comprehensive survey that provides details and examples beyond the scope of the overview provided here can be found in the publication by Kelleher and Pausch [68]. Both visual programming languages (which provide programming structures in visual form) or natural language programming languages (which feature syntax that is intended to be very close to written natural language) aim at providing programming to users who are not able to program in a regular programming language. Large collections of source code with abstract identifiers, complicated syntax rules and unintuitive error messages from compilers as well as the need to set up IDEs and build environments are hurdles for potential programmers that have to be overcome before a working program can be written. Another advantage that visual or natural language programming languages can provide is a better overview of what a program does.

Several authoring tools make use of such systems, including the e-Adventure authoring tool described above. Szilas [140] points out the appropriateness of graph structures for authoring of virtual character behaviors in the BECool system.

Resnick and Silverman [119] present the following set of criteria for didactically sound, motivating and user-friendly approaches to author support in this way:

- "Constructivism
- 'Low Floor Wide Walls'
- No enforcement of ideas
- Support for different work styles
- Simplicity
- Black boxes
- Basic understanding of programming
- What users want and not what they ask
- Develop applications in such a way as you want to use them



- Iteration”

As these criteria not only relate to non-programmer support but also overlap with the requirements for authoring tools, they are considered in the requirements analysis carried out in the following chapter.

### 2.5.1 *Natural Language Programming*

Natural language programming languages appear to users to be similar to natural languages, most often English. A comprehensive overview is provided by Knöll and Mezini [78]. The appeal of natural programming languages is the low threshold to learning them, since they resemble normal written text.

However, the vocabulary and grammar of a human is too complex and ambiguous to be understood completely by current IT systems. Therefore, systems that limit syntax to a construct resembling natural language currently are the best option in this field. This requires authors to learn the syntax of the language, which mitigates the initial advantage of natural language programming.

“Inform 7” [102, 117] is one of the foremost examples of natural language programming in the context of games. It is one of a line of programming languages for works of interactive fiction, with its predecessors being regular scripting languages. The language uses syntax that is easy to read for humans and rarely uses classical programming language concepts such as loops or tables. The target user audience of Inform 7 are the creators of interactive fiction and is therefore limited to textual input and output. Since it does not offer basic mathematical and logical functions, it is unfit for classical programming tasks from mathematics and engineering.

Inform 7 does not feature semantic analysis. However, errors are reported by the interpreter in natural language which is more intuitive than compiler errors for normal programming languages. Objects in the game can be described based on a set of basic classes and be connected via rules. The syntax and vocabulary understood by the language interpreter is fixed and cannot be varied as in normal human language. It therefore has to be learned by the author.

### 2.5.2 *Visual Programming*

Several approaches to visual programming exist. The differences between systems can be broken down using the same categories as those described by Bulterman and Hardman [20] for authoring tools. The category that we can refer to as structure-based is characterized by systems which offer structures that programmers can fill using drag & drop functions. As an example, Toyoda et al. [150] follow an approach in which applications are composed in a visual way from components. Input and outputs are connected via formulas or more graphical elements. Gaps in the resulting structures (referred to as “holes” by Toyoda et al. [150]) must be filled with formulas or parameters in order to achieve the intended result.

The analogue to the graph-based category of Bulterman and Hardman [20] are flowchart-based approaches. In these systems, individual states refer to transformation tasks linking the input values to outgoing transitions. See for example the survey by Johnston et al. [65] on dataflow programming. By manipulating the transitions and states and their conditions, complex behaviors can be visualized in a simple

fashion without the need to understand a programming language that is underlying the visual frontend.

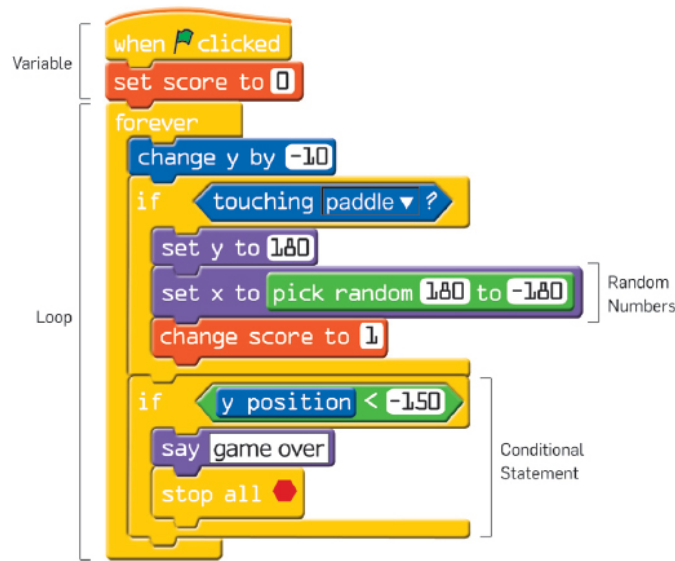


Figure 6: An example script in the Scratch programming language (source:[120]).

The Scratch system [120, 94] shown in Figure 6 is a programming environment intended for kids and adolescents. It uses shapes in order to visualize the structure of scripts. Each language element is represented by shapes that make it obvious in which ways they can be combined. Parameters can be selected via drop-down menus or be entered in the fitting slots of elements.

Another example for a visual programming language is the “Alice” tool [67] that is intended to teach programming concepts by allowing users to program their own game using a visual programming language. An extension of the tool, “Storytelling Alice” [69] allows programming using a hybrid approach between natural language and visual programming.

## 2.6 DEVELOPMENT OF EDUCATIONAL GAMES

In this section, we analyze the development processes of digital games in general and in comparison to educational games by surveying available accounts of game developers, best-practice reports as well as correspondences with game developers carried out during the conceptualization phase of this thesis. This examination leads to a process model of educational game development, which will be used as the basis for integrating existing workflows into the authoring tool conceptualized from the following chapter onward.

In order to assemble the information in this section, we interacted with the German educational game developer Braingame and carried out a literature survey, including the accounts by Amory [4], Hodgson et al. [56], Sommeregger and Kellner [135], Masuch et al. [96].

### 2.6.1 Digital Game Development

Digital games can have widely varying forms and are developed in a multitude of contexts (from independent teams or enthusiastic individuals to globally distributed teams working with hundreds of team members [29], therefore a comprehensive overview will always omit details. In the following, we assume that a finished game includes the following components, some of which are optional (e.g. a text-only game will not include visual elements)

- Game Engine
- Game Code (logic, game mechanic)
- Assets
  - Visual (textures, character models, animations, ...)
  - Audio (sounds, dialogues, music, ...)
  - Texts
  - Higher-Level assets (such as game levels combining multiple assets)

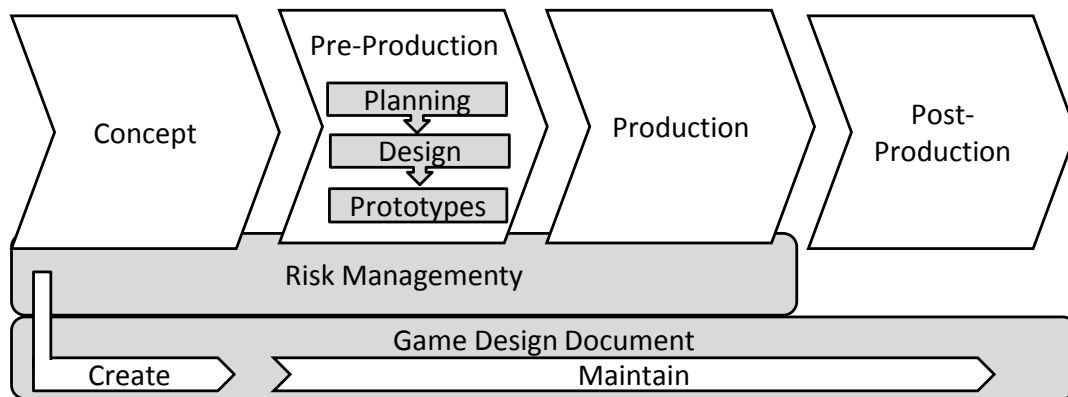


Figure 7: The typical phases in game development (adapted from [29]).

For the general phases of game development, we use the description by Cohen and Bustamante II [29] as visualized in figure 7. The production of a game (especially in commercial settings) is usually preceded by a concept phase and pre-production. During the concept phase, a vision and initial game design for the game is assembled along with a set of concept art which is used in pitching a game to an investor, usually a games publisher. During pre-production, the game design is refined and written down in the form of a design document. In narrative-driven games, the game's story as the means of binding the puzzles of the game together is written and fleshed out with the help of concept artists. When the genre of the game has been defined, designers can often make use of sets of conventions established in the respective genre. For example, most games from the genre of adventure games have very similar interfaces concerning navigation, dialogues and inventory management.

Additionally, a development plan is created and initial prototypes are used to investigate potential technical barriers and test gameplay early.

During the production of the game, the game's engine is either created or re-used from another source (a previous project, a commercial or freely available engine). Using the engine, programmers implement the game mechanics as specified during the game design phase. Assets (such as virtual character models, images, user interface elements or sounds) are produced based on concept art and the game design and are integrated in the game. This step can involve programmers again, since content has to be integrated with the logic of the game. Many game engines use scripting languages such as LUA or Python for programming at this level. This calls for a user trained in the tools necessary who is able to program in the scripting language of them.

Quality Assurance (QA) is carried out in parallel to the development of the game, with the goal of reducing technical and logical or content-related problems.

Management of game development studios and development methodologies are widely varied in the industry [11]. Many studios choose a hierarchical structure involving managers overseeing the studio and projects, a group of directors for the involved disciplines (e.g. a creative director for all creative aspects of the game) and teams of specialists (programmers, artists, animators, among others) who are assigned leads, usually the most experienced team members who organize and represent the team. Additional staff can be divided into development support carrying out quality assurance and studio support including organizational and human resources staff. Sizes of game teams can vary over the time of a project, for example the team size can grow during the production phase and shrink again during post-production.

Different development methods are used during game development. While the classical waterfall model as described by Royce [125] has been used widely, a growing number of teams employ agile development methods such as Scrum, where a game is developed during a set of development cycles instead of one large chain [110]. Teams using agile methods especially work with prototyping, where the design, gameplay, input methods and other variables of game development can be tested early in the development. For examples, see the accounts by Fullerton [45] and Ollila et al. [103].

Due to the highly varying roles and tasks in game development, different users have different workflows and tools. For example, story authors might use word processors for writing game documents or tools for non-linear writing. Artists use various 2D and 3D tools as well as specialized tools for tasks such as texture generation. Programmers use integrated development environments for writing code. For combining all assets into the final game, many studios develop in-house tools similar to game editors which are highly optimized and require intensive training and which are not offered to the public. In order to synchronize the workflows and avoid incompatibilities, game studios develop production toolchains, i.e. sets of guidelines which tools users should use along with the prescribed formats and conventions.

### 2.6.2 Educational Game Development

For the development of educational games, Tate et al. [142] state that the foremost focus of the development team should lie on the target group of the game. Therefore, this target group has to be defined before the remaining steps of the development are carried out, including factors such as age, previous game experiences and knowledge

of the target domain. This results in user-driven design and development, in which members of the target group are included in the development process at all stages.

New groups and roles of users are necessary for the development of educational games. This of course increases the complexity of the project and introduces new overhead and synchronization problems. The main additional groups added are domain experts, introducing their specialist knowledge about the target domain of the game, as well as pedagogues (especially for educational games) who establish the educational design of the game. The additional work carried out by these groups includes the creation of exercises or pools of exercises. In practice, this work is carried out in general purpose office tools and is disseminated to the rest of the team for integration into the game.

Apart from these added team members, the core game development groups also receive new tasks or task aspects in the development of an educational game. The work of game designers is influenced by the inclusion of domain experts, since the serious purpose of the game should be reflected already at the level of the game's design. For example, puzzles in the game should reflect the real-world practices that players learn while playing the game. In the case of narrative-driven games, the narrative could embed domain knowledge, by including it in dialogues or by assuring it's accurateness to the game's purpose, among other means. For example, an historical game's setting and plot should be true to the epoch it is placed in.

This necessity for close integration of domain knowledge into game production continues to have ramifications in all aspects of the game. Art production is required to produce assets which conform with the purpose and context of the game. Programmers are required to realize features that support the game's purpose directly (for example, by programming simulations of a domain) or indirectly (by developing general-purpose systems that entertainment games do not require, such as adaptivity).

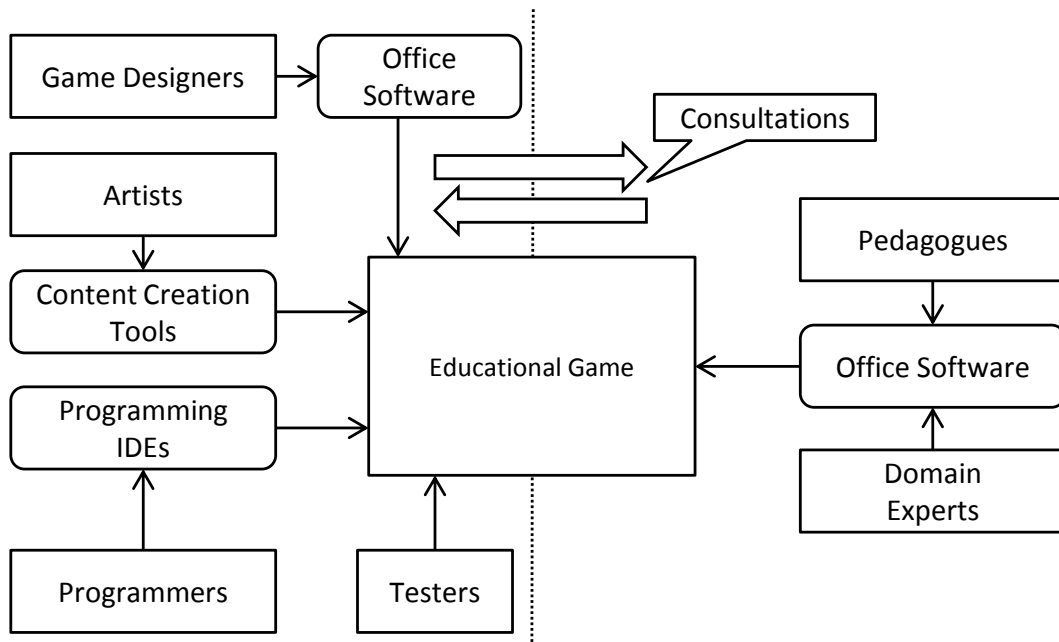


Figure 8: The separation between technical staff, game designers and domain experts.

However, even though we describe this close integration as necessary, in practice we however often find a separation between the technical realization of the game, the game design and the domain content, each characterized by different groups. The results of such a separation as seen in Figure 8 are often problems in communication and coordination between these groups over issues such as game mechanics and content, often aggravated by differing tools (office tools, content management systems as well as programming IDEs (Integrated Development Environment)) and the associated models and processes.

The following list provides an overview of the roles we have assembled so far:

- *Domain experts* contribute the knowledge about the target domain to the game. Initially, they are involved in specifying the requirements of the serious game, e.g. the knowledge to be learned or the intended training effects. During the course of the project, they contribute content in various ways, usually by providing it in a form that has to be integrated into the game by other members, e.g. artists or programmers.
- *Game designers* create a game design based on the requirements of the educational game on the one hand and the goal of providing a good game on the other hand, combining the two in the optimal case.
- *Game programmers* provide the technical basis in several areas of the project. Three major areas are the creation of a game engine that the game runs on, the creation of tools which are used to build the game and integrated assets, and finally programming customized elements not supported at the level of the game engine.
- *Artists* create concept art during the design phase of the game which is used to visualize the game and create a common vision for all team members. Later, they are tasked with creating the assets of the game. We include artists from all fields (visual, sound, modeling, animation and others) under this heading.

Adaptivity, as was seen above in the discussion of adaptive learning and adaptive games, is highly desirable as it can increase the efficacy of learning and the enjoyment of the game at the same time. However, adding adaptivity to an educational game leads to the introduction of another layer of complexity in the processes. The design (especially concerning the narrative and the game mechanics) has to be adjusted towards allowing adaptivity by providing several different ways to play the game or by different variations for the paths the players can take through the game.

Adaptivity can result in higher costs due to the need to create additional content or program extra game mechanics that allow the game to be adapted based on player preferences. On the other hand, contrary to content in non-adaptive games, no player will see all this additional content that has been produced.

Another challenge for designers and storywriters lies in the dynamic and algorithmic nature of adaptivity. The “narrative paradox” as described in section 2.2.2 captures this to a certain degree: in a classical, linear medium such as a movie, the consumer of the entertainment product has no freedom in choosing how the experience continues. However, a player in a game can influence the continuation of the game. This leads to the author of the game not being completely in control of the narrative, but having to foresee the possible actions of the player and providing

the game mechanics and story for each action. On top of this, adaptivity adds an element of uncertainty for the author since the actual continuation of the game depends on the current state of the user models, thereby potentially differing in each play session. Authors, especially those trained mainly with classical, non-interactive and linear media, can find it difficult to retain an overview of the whole game and the flow of events in the game with the player and the adaptive algorithm influencing it continually. This effect can similarly be observed in e-Learning [44]. For these authors, it is important to acquire a feeling for the effects of adaptive algorithms.

A special case in game development is that of re-releasing an existing game for new platforms or with new content. Since the focus of this thesis lies on providing an educational game authoring tool, we refer to this as “re-authoring”. We use this term in reference to the practice in e-Learning [118], where existing contents are refined and re-combined in an authoring tool to be used in new scenarios. The need for re-authoring is underlined by a review carried out with the German educational game development studio Braingame, in which it became apparent that common educational game development problems also apply for re-authoring. Furthermore, since the technical basis of the original game development is often not available, a new technology in the form of an authoring tool can be introduced here.

An authoring tool for games can help in this regard in two major ways: First, the structured authoring process can help a team to re-author the game in less time than a complete re-creation of the game would take, especially if the authoring tool already has implementations for the game mechanics of the re-authored game. On the other hand, re-authoring often entails enhancements over the original game that are necessary in order to reach both new audiences and those customers who already possess the original. An authoring tool can expedite the process of adding new content and can be used to add new features that are included in the authoring tool such as making a game adaptive.

## 2.7 CONCLUSION

Educational games have a large potential to be used successfully in education, since they combine the motivational aspects of playing with educational content. This combination however results in the challenge of providing games that integrate the aspects learning and playing to form a coherent and complete game instead of games that are separated into learning and playing aspects. The separation between game developers and context experts commonly found in the creation of educational games is one of the reasons for this separation. Therefore, an authoring tool for educational games has to account for all user groups included educational game development.

Adaptive games show the possibility of supporting the integration of learning and playing by adding methods that allow games to be optimized during play for specific players. While adaptivity is common in e-Learning, it is a relatively new field in educational games. An authoring tool for adaptive educational games has to provide support for challenges related to adaptivity, among them the need to train authors in understanding the effects of adaptive control of games and the need to support the creation of additional content required for adaptive games.

Concerning the production aspects of games, development has changed since its beginnings from workflow centered on programmers towards data-driven frameworks that are configured using game editing tools. However, such tools are commonly complex, require programming languages and are produced for professional game developers. Authoring tools are an alternative that has been demonstrated to be applicable in the contexts of multimedia and e-Learning. They often provide methods to allow non-programmers to create games, commonly using visual and natural language programming languages. However, authoring tools for educational games are a relatively novel field with few examples, especially concerning adaptive educational games. Therefore, a need for an authoring tool for this domain exists. In the following chapter, the challenges and requirements assembled in this chapter are used to identify three possible application scenarios for such an authoring tool.





**B**ASED on the identified needs from the state of the art analysis in the previous chapter, we now provide three application scenarios in which an authoring tool can be introduced in order to facilitate the creation of educational games. From an analysis of these application scenarios we specify a set of requirements an authoring tool has to meet. Using these requirements, we provide an overview of the concept for an authoring tool for adaptive educational games.

### 3.1 APPLICATION SCENARIOS

We continue with the definition of application scenarios for the authoring tool conceptualized and developed in the course of this thesis. The application scenarios and the subsequent requirements analysis are based on the observations of the general workflow for the creation of digital (educational) games in the previous chapter, interactions with game developers as well as potential users in the context of the user requirements study in the next section and the open community of the authoring tool StoryTec (cf. appendix [A.1](#)).

We have identified three application scenarios in which users can profit from a dedicated authoring tool for educational games:

- *Individual users creating a game.* The prototypical case for this application scenario is a teacher who intends to create an educational game for a class he or she is teaching. Similar to universities where lecturers are able to create the e-Learning material for their courses by themselves using authoring tools for this purpose (cf. section [2.3.1](#)), this teacher should be empowered by the authoring tool to create this game based on his or her intended learning matter and available content.

Currently, individuals who are in this situation have few choices of how to proceed with the task of creating such a game. The available game authoring tools are often complex due to being developed with professional game developers in mind. They often require knowledge of programming languages, which we cannot assume in this scenario. Furthermore, users in the scenario require assistance with game design and development since they are not trained game developers.

A user study carried out with teachers presented later on in this chapter corroborates this application scenario. The diagram in figure [9](#) displays the envisioned use of the authoring tool in this scenario.

- *Small teams creating a game.* The second application scenario we analyze is that of a small team creating a game. This could be a setting in which the training department of a company is tasked with the creation of a game to be used in-house, or a small educational game developer creating a game. We limit this scenario to small teams, as the large size of some current game projects with

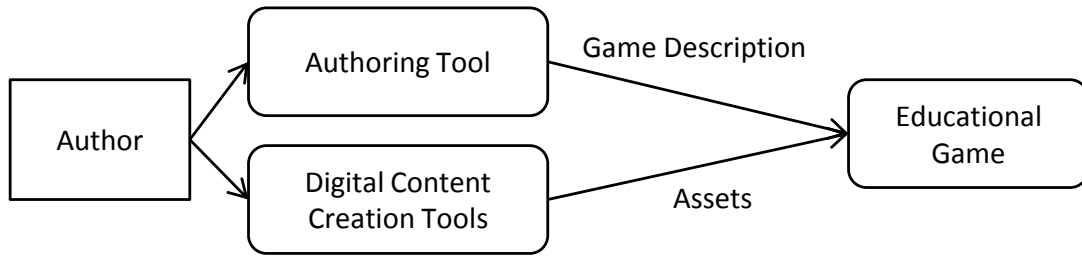


Figure 9: Application scenario 1: A single author creating an educational game.

more than hundred persons working on a game would require support for collaboration outside of the scope of this thesis.

In this use case, we find the groups and roles of users identified in section 2.6, notably also including programmers. However, this application scenario suffers from the effects introduced by multiple users with different backgrounds collaborating. Two negative effects can be observed. The interaction between technical staff, creative team members and domain experts can lead to latency times, which increase the time required to produce the game. The second negative effect lies in the loss of game content which results from a missing common understanding between domain experts and the rest of the team, resulting in effects such as game design that is incompatible with the educational purpose of the game or learning design that cannot be realized with the technical capabilities of the project. These effects are worsened by the use of a multitude of tools that are incompatible to each other.

An extension that included in this thesis is that of re-authoring of an existing game. In this case, a game that has been created previously with the normal game development workflow is to be re-created in an authoring tool. There are several motivations for this, including the need to port the game to new platforms it was not developed for initially or a new release of the game with minor adjustment is intended, but the original development workflow is not practicable anymore (e.g. no specialists from the original team are available for the development). Figure 10 displays this scenario, with all involved users sharing the authoring tool in order to carry out their work in the production of the game.

- *Individual users customizing a game.* The third scenario for an authoring tool is that of individual users who customize or personalize a game that has been previously created. This scenario is similar to the first one, but is differentiated by the factor that the game has been created in an authoring tool and is only configured by a domain expert. The concept of collaborative authoring is the foundation for this scenario. In a first step, an author who is trained in the use of the authoring tool creates the structure of the game. In a second step, a domain expert provides content by integrating it in the provided game. The author who is customizing the game for the needs for specific end users should not be burdened with issues related to the creation of the game, but instead be able to focus on changes or settings that benefit this end user. An example is a teacher creating a game for a class based on a preconfigured game, personal-

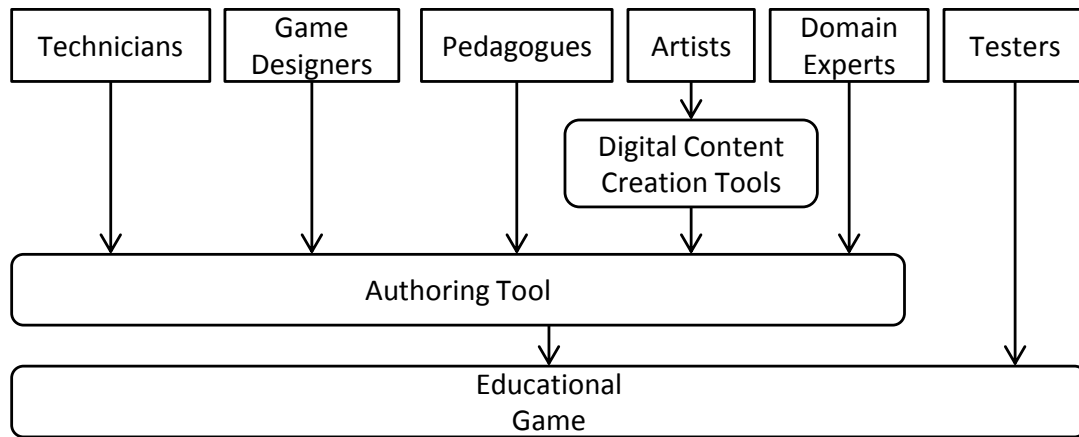


Figure 10: Application scenario 2: A small team creating a game.

izing the game to the specific needs of the class such as the curriculum being addressed.

Figure 11 provides an overview of this scenario.

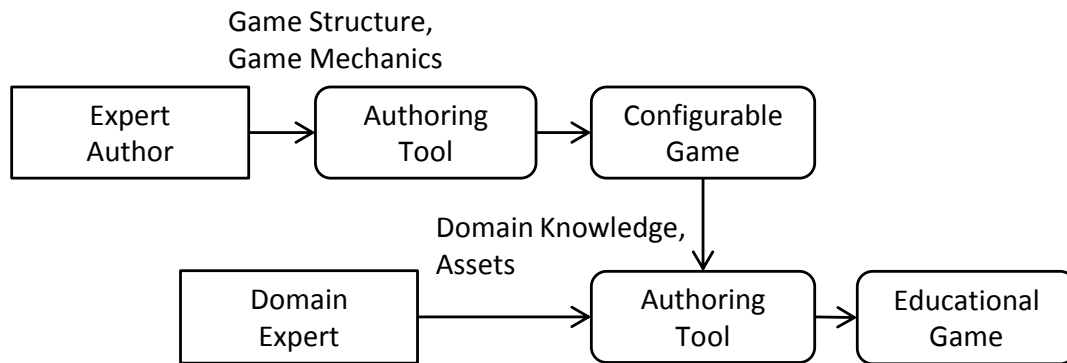


Figure 11: Application scenario 3: Individual users customizing a game.

### 3.2 USER REQUIREMENTS STUDY

In order to determine the needs of the target users better, especially concerning the group of teachers that are addressed mainly in application scenario 1, a study was design and carried out. The study included interviews with teachers, before which a short presentation on the uses and capabilities of authoring tools was presented to explain the concept of authoring tools to the participants. Afterwards, the participants were given a questionnaire created for this purpose which included items for assessing demographic data as well as several items for measuring the needs and requirements of teachers regarding educational games and authoring tools.

$N=51$  teachers were recruited for this study. The teachers questioned mainly teach at primary schools (41%) and schools for special education (41%). In the following, the we focus on the analysis and discussion of the questionnaire results.

From the answer to the question “please rate your computer skills”, we can see that the target user group is in fact largely composed of non-programmers (see figure 12). Noteworthy is furthermore that almost all teachers claimed to use office tools for

preparing their lessons (50, 98%), followed by graphics software (13, 25%), worksheet generators (9, 18%), smartboard-software (5, 10%) and other tools (4, 8%).

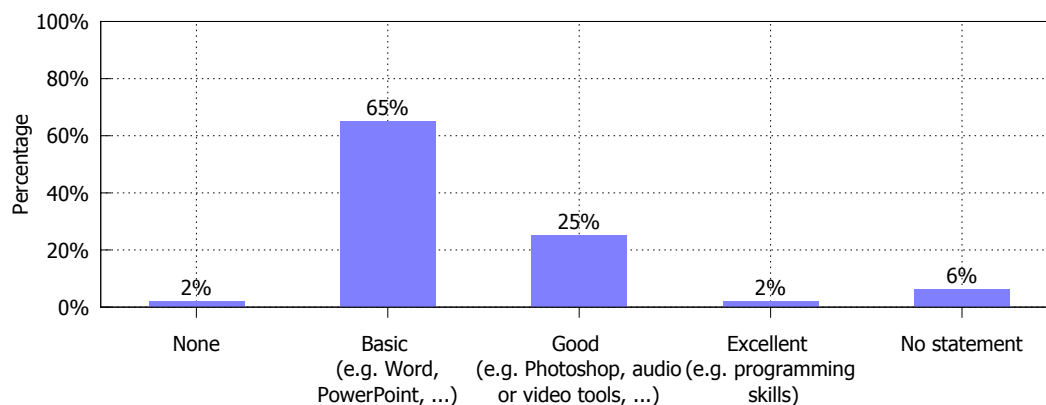


Figure 12: Answers to the item “Please rate your computer skills”.

Concerning the use of educational games in the classroom, the teachers tended to agree to the statement that students can achieve media competency from educational games, which supports their use in the classroom. On a scale from 1 (completely disagree) to 4 (completely agree), the teachers answered with  $m=2.92$  ( $SD=0.69$ ) on average. The statement that educational games motivate students far better than normal classroom activities was rated with an average of  $m=2.78$  ( $SD=0.65$ ).

However, even though the teachers saw the use of educational games positively, a need for appropriate educational games was identified that is not met by the available commercial educational games. The questionnaire item “I can find an educational game for all topics I teach” was rated on average with only  $m=1.88$  ( $SD=0.57$ ).

The teachers did not have experience with creating their own games. 47 (92%) have never attempted to create a game, and only 3 (6%) claimed to have successfully tried it. As figures 13 and 14 show, the teachers tended to think that they would be able to create educational games themselves ( $m=2.2$ ,  $SD=0.88$ ) and claimed an interest in doing so using an authoring tool ( $m=2$ ,  $SD=0.83$ ).

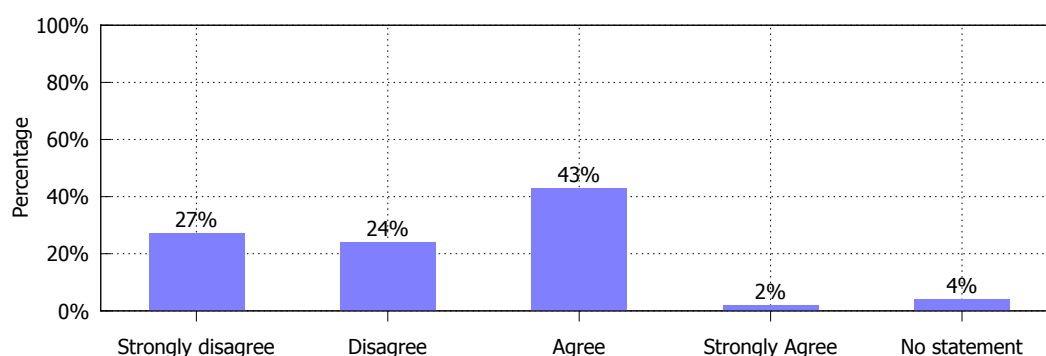


Figure 13: Rating of the statement “I am able to create a good, didactically sound educational game”.

As a final result of the user study, we see the answer to the question “how much time would you invest in a game that students player for 15 minutes” (figure 15) indicates only a short amount of time teachers would be willing to commit to creating

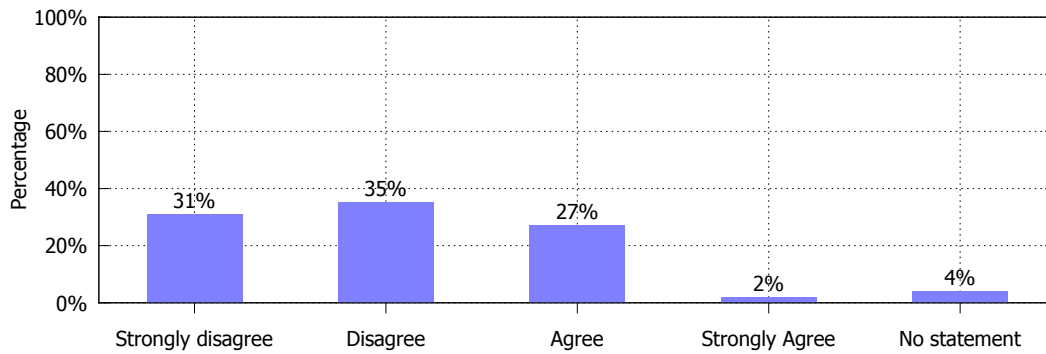


Figure 14: Rating of the statement “I wish to create educational software with an authoring tool.”.

such a game, with the answers “up to 30 minutes” and “up to 60 minutes” being chosen most often with 26% each.

The results of this study are used in determining the requirements in the following analysis.

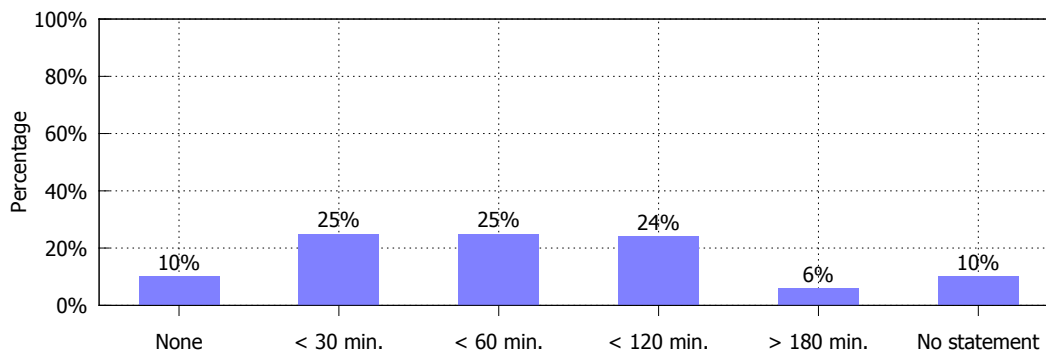


Figure 15: Answers to the question “How much time would you be willing to commit to authoring a game played by students for 15 minutes”.

### 3.3 REQUIREMENTS ANALYSIS

When identifying requirements for an authoring tool, it is important to realize that all potential users have different backgrounds and use different tools in their lines of work.

Programmers utilize IDEs for the programming languages with which they create the technical basis of the game. However, most other members of a game development team are characterized by limited programming skills. Artists and game designers often have a design background and have knowledge of tools such as Adobe’s Creative Suite<sup>1</sup>. Therefore, the design of an authoring tool integrating these users should consider typical concepts underlying these tools, which are reflected in the interaction design as well as layout and functions of GUI elements.

In contrast, domain experts and writers often have none or only rudimentary knowledge of such systems, but are familiar with office tools and use them in their daily work, for example by providing lists of exercises for educational games in the

<sup>1</sup> January 24, 2013] <http://www.adobe.com/products/creativesuite.html>

form of spreadsheets or storyboards. Similarly, the requirements study for teachers in the previous section showed that teachers often use office software for their lesson preparation.

When integrating users with such diverse backgrounds and expectations into a single authoring tool, it is necessary to find a compromise that is appropriate for each user group, including an alternative to programming languages commonly used in game editors.

In the following, we assemble a list of requirements from the above application scenarios keeping the diverse backgrounds of users in mind. Many of the requirements identified overlap and are therefore required in all application scenarios.

- *Individual users creating a game.* In this scenario, we can identify the following requirements:

*No programming.* Since the users we assume lack the knowledge of programming languages and do not have the time or budget to learn them, we cannot use traditional programming languages for interactive elements in the game. Instead, mechanisms have to be identified and included in the authoring tool that allow users to configure non-trivial game logic while not requiring an understanding of a programming language.

*Cost-effectiveness.* An author in this scenario has a very limited or no dedicated budget for creating a game. Therefore, the workflow of the authoring tool has to support a cost-effective way of creating a game. This can be achieved with the tool supporting the author in spending as little time as possible to create the game. Furthermore, the choice of game genres, leading to varying requirements on game content, has an influence on this. For example, 2D content is cheaper and more available to the users we assume in this use case than 3D content.

*Quick to learn.* Since authors in this use case are only working on a game in the authoring tool in a limited timeframe, the authoring tool should be easy to understand and support learning in order to minimize the time it takes for an author to commence working (cf. figure 15 from the user requirements study for teachers in the previous section).

*Support for easy content integration.* Authors in this use case do not have artists supporting their work and do not have the budget to create or pay other parties to create content. We can meet this requirement by making content integration easy and align game genres and interaction in the authoring tool towards available content (e.g. already existing images and videos).

*Small-scale authoring.* In this case we assume that the user will create games on a relatively small scale. Therefore, the authoring tool should scale functionality to this purpose and allow quick authoring of relatively small games.

*Cross-platform publishing.* Taking the use case of a teacher creating a game, we can see that distributing the finished game to the target audience (e.g. a class) is difficult. Schools still have restricted access to current hardware, a problem aggravated by restrictive security settings on available PCs. Therefore, it is worthwhile to react to the growing number of young people equipped with current PC hardware or mobile devices such as tablets or smartphones. In

order to make this feasible in this use case, the authoring tool should be geared towards cross-platform publishing that is transparent for the author.

- *Small teams customizing a game*

*Collaboration.* As was pointed out in the description of this use case, the main problems in this use case are introduced by multiple users working together and the associated communication overhead and resulting losses of information. A way to counter this lies in planning a collaborative authoring tool which is equipped to allow teams to deal with this problem.

*Integration of all users/roles.* In a similar vein as the previous requirement, we argue that an authoring tool should integrate all users and roles in game development. This means that the workflow of the authoring tool encompasses steps in which all members of the team can add to the project in a well-defined way.

*Support for re-authoring.* Re-authoring can be a practice that teams of game developers can carry out. An authoring tool can support this practice, for example by allowing content from the existing game to be integrated easily and by further added value such as cross-platform publishing.

*Cost-effectiveness.* The authoring tool should allow a very cost-effective way of developing games, as educational developers commonly face smaller budgets and have smaller target audiences than entertainment game developers. This can for example be achieved by fostering re-use and a decrease in the number of required experts.

*Cross-platform publishing.* With the multitude of hardware platforms and markets available to users today, it is important to consider cross-platform publishing. Especially with educational games, with typically lower revenues than entertainment games, it is imperative to deliver the game to all interested users regardless of the device they use without additional overhead for each device. Furthermore, target users of educational games will typically not be willing to buy a new device in order to play the game.

*Large-scale.* Games created by a team of users will be larger than those created by an individual. Therefore, the authoring tool should support the creation of large games. This entails that performance does not drop significantly when larger structures are created on the one hand and that authors are supported in keeping an overview of the structure and logic of the game despite the size on the other hand.

- *Individual users customizing a game*

*Collaboration (with game creator).* In essence, this scenario is a collaboration between two parties: the initial developer of the game, and the individual user who customizes it using a set of parameters. Therefore, mechanisms for allowing a collaboration between these two parties should be included.

*Quick to learn.* The users in this use case are characterized by a lack of time to learn the use of an authoring tool. Therefore, they have to be supported in learning the essential functions that they have to use as quickly as possible.



| REQUIREMENT               | SCENARIO      |         |                  |
|---------------------------|---------------|---------|------------------|
|                           | 1: INDIVIDUAL | 2: TEAM | 3: CUSTOMIZATION |
| NO PROGRAMMING            | Yes           | (Yes)   | Yes              |
| COST EFFECTIVE            | Yes           | Yes     | (Yes)            |
| QUICK TO LEARN            | Yes           | (Yes)   | Yes              |
| CONTENT INTEGRATION       | Yes           | Yes     | Yes              |
| SMALL SCALE               | Yes           | No      | (Yes)            |
| CROSS-PLATFORM PUBLISHING | Yes           | Yes     | (Yes)            |
| COLLABORATION             | Yes           | (Yes)   | Yes              |
| INTEGRATION OF ALL USERS  | No            | Yes     | (Yes)            |
| RE-AUTHORING              | No            | Yes     | No               |
| LARGE SCALE               | (Yes)         | Yes     | Yes              |
| GEARED TOWARDS THE TASK   | Yes           | Yes     | Yes              |

Table 4: The user requirements drawn from all use cases.

*Geared towards the task.* The configuring author only seeks to manipulate those settings that have an influence on the person he or she is configuring the game for. Therefore, the authoring tool should be configurable itself so that users who do not need the full set of features are not burdened with all of them.

*No programming.* Since domain expert users cannot be expected to be able to program (similar to the first use case), we include this requirement here as well.

*Content Integration.* One aspect of the configuration work users in this use case carry out could be that of adding suitable content to the game. Therefore, content integration should be as easy as possible for them. This also includes the exchange of elements with other elements, which can change the character, aesthetics and players' perception of the game while keeping the game mechanics in place. Therefore, different user groups and player types can be addressed by such exchanges of content.

Table 4 summarizes the relationships between the requirements and the scenarios. The requirement *Geared towards the task* is included for all scenarios as a basic requirement of authoring tools in general. This indicates that the authoring tool should provide specific support, for example for the authoring of educational content.

### 3.4 CONCEPTUALIZATION

The analysis of the development of educational game development carried out so far has shown the large potential of educational games and the need for authoring tools for adaptive, educational games. The requirements such an authoring tool has to meet have been drawn from an analysis of three application scenarios. Based on this need, we provide a concept in the following that combines the ideas found in the

related work in a novel way in order to offer an authoring workflow which allows authors to work on adaptive educational games. Before details of the concept are covered in the following chapters, we present an overview of the concept to allow the reader to see the connections between individual parts of the concept.

The overarching goal of this concept lies in an improvement in the workflow and communication between various members of development teams and the support for non-programmers. The design of a tool to be used in all aspects of the game development process is explicitly not the goal, since there are specialized tools such as 2D image editors, 3D art tools or integrated development environments for programmers. Nevertheless, the concept of the authoring tool should be open for input from all these tools in the form of assets or components which are integrated into the authoring process.

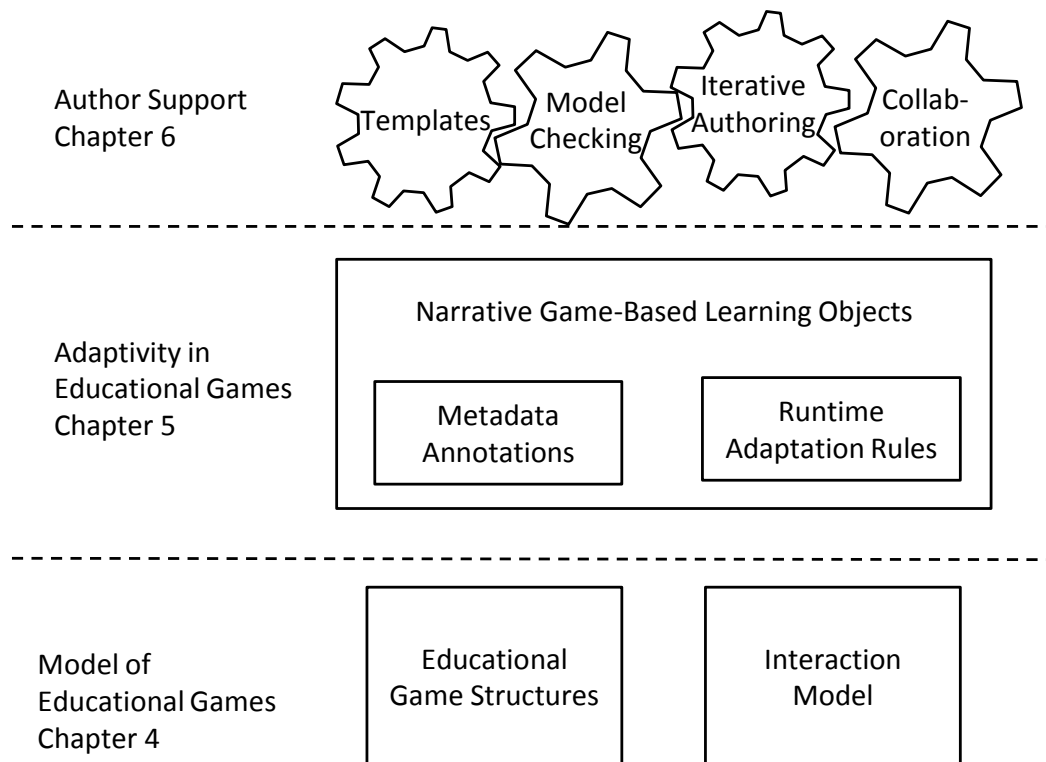


Figure 16: An overview of the aspects of the authoring tool concept described in the following chapters.

Figure 16 provides an overview of the layers contributed by each chapter in which the concept is expanded upon. The main aspects of the concept are as follows:

*Educational game model.* The basis of the concept is laid by the game model defined in chapter 4. The motivation for this initial step lies in the goal of providing a structured model on which the envisioned authoring process can be built. Furthermore, if the paradigms of visual programming and templates are to be part of the authoring process, we require a well-structured model as a basis.

This model defines the basic units of a game. Temporally and structurally, the game is broken down into *scenes*, which represent small parts of the game (e.g. one screen). Scenes are connected to each other via *transitions*, indicating a possible change of scene in the game. *Objects* are defined to populate scenes with items,

characters and other game objects. *Parameters* are used to configure scenes and objects.

The game description model is utilized throughout the authoring tool. As a foundation, it allows the authoring process to be well structured by providing a framework of hierarchically organized containers and objects which can be used to construct games from many genres. More concretely, this structure can be visualized and be used to edit the structure of the game using a graph representation, affording a high-level overview and control of the whole structure of the game.

*Structured authoring.* The game model is used in the remaining concept to allow a structured process of authoring. As was laid out above, the creation of games is inherently more complex than other multimedia applications and is often approached in a quite unstructured way. In a general-purpose game creation tool, authors would have to assemble the structure of the game themselves by building the foundational game mechanics and methods for content integration themselves. Working with a model as the one described in this concept allows authoring on a higher level of abstraction, where basic structures are provided and authors only have to carry out necessary modifications. This, in conjunction with other mechanisms such as visual programming, enables novice authors and non-programmers to work on a game.

When such a model-driven approach is chosen, the effects of changes to the model should be made clear to an author at all times during the authoring process. This includes a description of affordances the model provides to an author, using various methods and media. For effects on the gameplay, a preview in the form of a video trailer showing typical results of changes can be shown. While working on the game in the authoring tool, the author can be supported by pointing out modifiable aspects of the model.

*Interaction.* Interaction in games is realized by applying concepts from visual programming languages and model-driven development. Instead of giving authors basic control of all aspects of the game, they are instead provided with high-level commands. Commands can be structured sequentially or in parallel, indicating sequential or branching execution. Branches can be chosen and execution interrupted in a sequence by adding Boolean conditions. The resulting command trees are triggered by high-level events from the game, termed stimuli.

*Templates.* Based on the model of games, we define structural and interaction templates. Structural templates exploit the model by providing pre-configured parts of games that can be re-used and which can give structure to authors who are new to game authoring. Interaction templates encapsulate complex game mechanics by providing an easy interface to authors in the authoring tool and an implementation of the gameplay in a game engine.

Especially interaction templates will be shown to be beneficial towards the goal of providing an authoring tool for non-programmers. In a [WYSIWYG](#) editor used to configure how the game will look and play, authors can be provided with a suitable and easy visualization of the interaction template. In this way, they are only concerned with the relevant information for their level of game development expertise. Templates add another layer of structure to the authoring process, since they can be used to provide best-practice examples to authors. For example, an author who is untrained in writing game narrative can be assisted by providing templates that include the structure of successful story models.

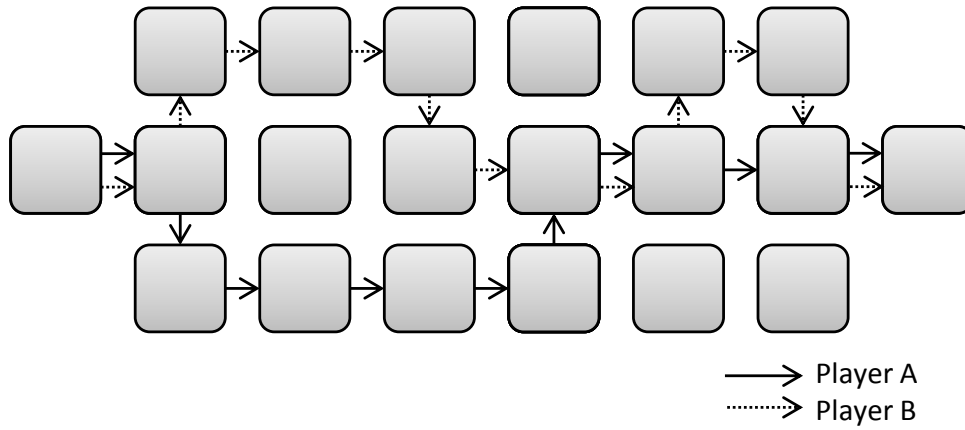


Figure 17: The structure of an adaptive game. Two players with different needs or backgrounds experience the same game differently due to multiple learning paths through the game.<sup>2</sup>

*Adaptivity.* Adding metadata for the purpose of adaptivity to scenes allows us to create games that are adapted to a user during runtime. We provide this capability by extending the model of scenes to Narrative, Game-Based Learning Objects, indicating that adaptation is based on storytelling models as well as player and learner modeling. Adaptivity is directly built into the model of games, utilizing the existing structures of scenes, parameters, transitions and action sets. As Figure 17 indicates, an adaptive game can be modeled as a graph structure which is traversed at runtime not solely based on the choices by the player, but also by the control of an adaptive algorithm.

The primary purpose of adaptivity in the described system is to provide the positive benefits of adaptation as shown in the overview in chapter 2. Apart from providing a fixed path through a game, the system can adapt to the user and branch accordingly or even navigate the user through a pool of scenes based on storytelling, temporal and structural restraints. For authors, providing a framework for building a curriculum in the authoring tool adds a new step to the workflow which supports the learning design of the game and helps authors in deciding and modeling which knowledge they wish to impart to players. The combination of a model for games and author support mechanisms leads to a workflow for the development of games using the authoring tool described in this thesis which has clearly defined stages and roles for all involved users.

*Model Checking.* Based on the structured model underlying the authoring process, we can use model checking to find syntactic and semantic errors (such as scenes that can never be reached in normal play due to logic errors). Inspired by model checking in related areas, we describe a set of possible queries that can be made based on the game model and techniques such as data-flow analysis that can be applied in order to answer them. Model checking becomes especially important in larger games where authors might lose the overview due to the sheer number of possible states a game can be in, a factor aggravated in the creation of adaptive games, where authors have to yield some of their control over the game to the adaptive system.

*Collaborative Authoring.* The requirement for collaborative authoring is satisfied by expanding the game model with support for different users groups and a management of access options concerning aspects of the model. With this expansion, authors

can be separated according to their expertise, with the experienced authoring tool users creating views for novice authors or domain experts that expose fewer features of the authoring tool and which allow these experts to work on the aspects of the game relevant to them while being able to ignore the unrelated aspects of the game.

*Iterative Development.* The game model incorporates support for the practice of rapid prototyping, which is based on building complete versions of a software product, with the goal of eliciting early feedback that can influence the remaining development. Since interaction templates and scenes can be filled with placeholder content and game units be skipped by directly triggering the corresponding high-level events, prototypes can be built early and be played in applications specifically developed for testing purposes. Software wizards can assist novice authors in correctly authoring games.

The authoring tool concept described here builds upon prior work carried out in the Digital Storytelling group at the Computer Graphics Centre in Darmstadt. The game description model described in this thesis is based on similar assumptions as the story description language [ICML](#) of INSCAPE (cf. section [2.3.2](#)), realizing a hierarchical network of scenes connected by transitions. The concept of events is realized in both systems in the form of stimuli. Both systems share the idea of a Story Engine for interpreting the description model during runtime. Finally, the concept of a visual, abstract editor for the application structure in the Story Editor is shared.

The work presented here can however be clearly delineated from prior work. The shift from interactive storytelling to games leads to necessary changes and innovations in all areas. A new description model has been created to meet the requirements of adaptive educational games, especially by introduction of adaptive game elements and the Narrative Game-Based Learning Object ([NGLOB](#)) model. INSCAPE featured neither a visual programming approach nor the interaction template mechanism described here and, in fact, complex applications required use of the LUA programming language. The Story Editor component, initially the only re-used component in a completely new software framework, was soon replaced by a newer version due to usability issues as described in chapter [8](#).

### 3.5 CONCLUSION

This chapter served as the link between the related work in the field of authoring of adaptive educational games and our authoring tool concept. For this, the outcomes of the state of the art analysis have been consolidated as three application scenarios and a set of requirements. The three scenarios that have been identified are that of a single author (for example, a teacher) creating a game; a small team of users with various roles collaborating on an educational game; and a user who customizes a previously created game to the needs of a specific player. The identification of scenarios and requirements has been supported by interactions with actual users, especially in the course of a user study showing the needs and requirements of teachers concerning authoring tools.

This requirements analysis and the discussion of game production processes have been taken into regard for the elaboration of a concept for an authoring tool to be used in the creation of adaptive educational games. The ideas and methods from the fields multimedia, e-Learning and game authoring presented in chapter [2](#) along with

user support mechanisms, mainly visual programming, have been combined to form this concept. The basis of this concept is a game model including aspects for the organization of game structures as well as game logic. The game model serves as a foundation for several authoring tool features including support for non-programmers and structured authoring. The authoring of adaptive games is specifically supported by the extension of the game model to include adaptive mechanisms and metadata. Finally, several requirements including a short learning time, collaboration as well as cost-effectiveness are met by building specific user support mechanisms on the basis of the game model.

In the following three chapters, the salient features of the concept are described, beginning with the game description model.



THIS chapter lays the foundation for the concept by specifying how the structure and the logic of a game are to be modeled. This step of modeling a game is made in order to provide authors a clear and decisive way of entering the game content. The building blocks of the model are utilized in the following chapters to construct higher-level functionality of the authoring tool and have major influence on the organization of the implemented authoring tool, as will be seen in chapter 7.

Based on the discussion of game models in section 2.4, we choose the following basis for the educational game model:

- **Storytelling-based:** We pursue a storytelling-based model since we perceive it as that which allows the easiest and most versatile authoring process. Since games can be broken down into very small units which exist relatively independently (similar to learning objects), authoring can happen locally instead of globally. Other approaches, such as character-driven, are less transparent for authors (e.g. the way in which the game plays is not easily seen from the input files) and require more global structures which influence the whole game. As noted above, we therefore exclude pure games which have no aspect of storytelling.
- **Data-driven:** Since one of our goals is to provide authoring for non-programmers, the approach has to be data-driven. Otherwise, authors would have to manipulate the logic of the game by manipulating a programming language.

After motivating the choice of a data-driven model based on storytelling concepts, the model of game structure, content and logic is presented in the following sections. This model is based on the atomic unit of *scenes*, which are be connected via links referred to as “transitions”. “Objects” are placed inside scenes to realize scenery, characters or logical objects such as variables, and “parameters” used to control properties of scenes and objects. Game logic is configured by high-level commands, referred to as “actions”, which can be organized sequentially or branching and be conditionally executed. The execution of a sequence of actions is triggered by an event from the game environment, termed a “stimulus”.

## 4.1 GAME STRUCTURE MODEL

### 4.1.1 *Scene*

The concept of a scene is chosen as the atomic unit of a game’s structure in the model established here. This choice results from the overall choice of using a storytelling-based model as the basis for our game model.

In theater or movies, a scene comprises a small part of the overall narrative, delimited in space by the set or the theatrical scenery, with a set of fixed scene items (props)



and a set of actors. Similarly, in this concept, a scene is intended to model a small part of the overall game, containing all objects and logic to capture the interaction of the player while the scene is experienced by the player.

By choosing this model of a scene, an inherent inclination towards games with a narrative and with gameplay that fits well into being broken down into a set of sequential scenes is introduced. Due to this, games which are centered around gameplay and less on narrative will be harder to model. For example, the authoring of a 2D platform game will hardly benefit from the scene concept, since the smallest level of granularity that makes sense is that of a level. Therefore, if one intended to model this kind of game with the model presented here, each level would correspond to one scene in which all game logic would have to be embedded. Another case would be seamless worlds as described above which commonly feature quests. Here, the modeling based on scenes could be applied to quests, with scenes being used to model individual steps of the quest. While this approach to modeling such games is feasible, it is out of the scope of this thesis.

One modification to the concept of scenes as derived from linear and analogue narratives is the hierarchical organization of scenes which is allowed in this model. The structure of scenes is therefore a tree. The parent-child-relation in this tree structure indicates that a scene inside a sub-tree should inherit certain traits of scenes that are above it in the tree. This applies mostly to the objects contained in the parent scenes. When a scene is displayed, apart from all information contained in the scene itself, also all information contained in parent scenes should be displayed. Similar to the concept of a level in games as described above, this allows authors to place objects that are common to a set of scenes in the parent scene instead of adding them in each scene individually. A prime example is a set of scenes that feature the same background but differing actors visible.

#### 4.1.2 *Object*

Again drawing upon the metaphor of theatre and movies, scenes act as containers for objects. While a theatre scene would have the scenery, the props and the actors as physical “elements”, for the model presented here, the overall designation “object” is used for all elements a scene can have. Therefore, it is the umbrella term for any objects that are visible to the player in the game (such as images, 3D models or virtual characters), other media elements (such as sounds) or interactive elements (such as buttons or text fields). Apart from these objects the player interacts with directly, we have chosen to add also logical and control objects under the term “object”. These objects include mainly variables (for keeping track of information about the game state) and other objects not directly perceivable by the player since they control the game flow rather than the visual look of the game.

As has been noted above, when scenes are placed in a hierarchy, objects higher up in the inheritance tree will be inherently added to the lower scenes when they are shown to the player. For example, the highest scene could define a background, in front of which various objects are placed in the scenes that inherit from the parent scene.

### 4.1.3 Transition

In order to model the connections between scenes, the concept of a transition is introduced. A transition is a directed relation between two scenes, indicating that the game can jump from the scene at which the transition starts to the scene at which it ends. The conditions for this transition are defined below in the context of actions. Therefore, by adding transitions to the hierarchically organized scene structures, we arrive at a story model that is realized as a directed graph structure.

The choice for explicitly modeling transitions instead of only adding them implicitly is made for several reasons. First, it allows a workflow where the author can first add scenes, then think about the possible connections between the scenes and then continue to add the logic for the transitions. The alternative would be to define transitions implicitly, i.e. only when the author has specified a jump to a different scene to take place, a transition would be created. Since the explicit approach allows authors to first define the game structure and then fill it, we have chosen this alternative.

Transitions, can also be marked as being “free”, indicating that the exact transition that will be used to exit a given scene is chosen at run-time based on the adaptive choices in the game. See the following chapter for more details.

### 4.1.4 Parameters

In order to configure scenes and objects and to add metadata (utilized for adding adaptive features to the model), scenes and objects feature parameters. Scenes feature mostly metadata, including the average time an author expects players to remain in the scene during normal play, as well as the metadata for adaptivity. Objects make profound use of parameters for setting up their properties and control parts of their behavior. For example, for an image object, the image file to load is specified as a parameter.

Figure 18 shows an example game model with five scenes, one of which is nested in another scene. The scenes are aligned in a branching structure of transitions and feature objects and parameters.<sup>1</sup>

The following, extendable list of parameters is currently included in the game model:

- Boolean
- Color
- Composite
- Enum
- File
- Float
- Scene
- Skill

---

<sup>1</sup> In all figures of this thesis, model elements, especially scenes and transitions, are drawn in the same way used here in order to separate them from other visual elements.

- Stimulus
- Object
- String

Noteworthy parameters are the scene, skill, stimulus and object parameters, which are used to create links between elements of the model. For example, a specialized type of hotspot can be linked to the target scene it switches to, thereby allowing authors to choose the target scene directly by setting this parameter.

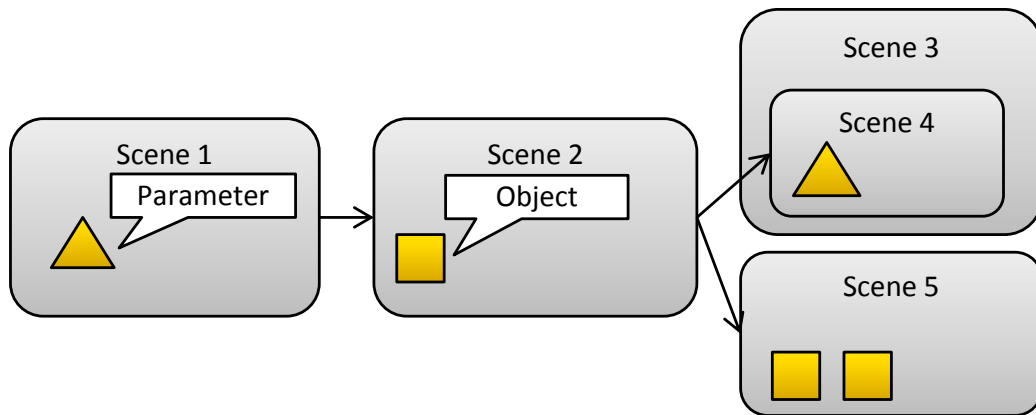


Figure 18: An example game structure with scenes, transitions, objects and parameters.

#### 4.2 GAME LOGIC MODEL

Using the concepts explained so far, only a static game without changes of scenes would be the result of the authoring process. Therefore, it is necessary to address the need for the author to control the game logic and issue commands to the game. This aspect of the authoring process is explicitly modeled here as it is one of the major requirements upon the model to allow non-programmers to configure the game logic with it. If this aspect was deferred to a script language which could control objects and transitions, this requirement would be violated.

Concerning the alternatives of visual and natural languages to use for authoring interactivity (cf. section 2.5), the visual language option is chosen here. This is due to several reasons. First, the structural model described above is intended for the graph-based authoring paradigm. If the interactivity was realized with a natural language programming language, a mixture of two paradigms would be the result in the authoring tool. Furthermore, current natural language programming still requires authors to remember the correct syntax, conflicting with the goal of providing an authoring tool with a low threshold for use. Therefore, a model supporting graph-based authoring of interactive elements is described here.

The goal the model aspects for adaptivity are the support for visual programming in the authoring tool implementation, which influences the choices made in this regard.

#### 4.2.1 *Action Set*

This concept uses the idea of modeling the game logic similar to the way the game structure is modeled. For this purpose, two basic patterns from programming languages are adapted, that of conditional execution and that of commands or functions. The basic units of game logic in this model consequently are actions, which encapsulate any kind of command that can be executed in the game, and conditions, which control whether a sequence of actions is to be executed.

Actions are intended to be of a significantly higher level than a function call in a programming language. An example which illustrates this is an action executed on a virtual character directing the character to move to a certain location on the screen. In a regular programming language, this would entail setting a path between the old and new position of the character, moving the character on the path by animating its position as well as starting an animation on the character. Instead of these details being handled by the author, they are encapsulated in one action.

The combination of actions and conditions is referred to as an “action set” in the model. Actions can be arranged sequentially, indicating that one action is executed after the previous one is finished. Before each action, a condition can be placed, resulting in the actions below to only be executed if the condition is fulfilled. By arranging actions in parallel and adding a condition before each block, a branching structure can be realized, in which only one of the branches will be followed.

In order to give details about the intended results, actions also feature parameters. These parameters control the details of the actions’ execution in the game. Re-using the above example of moving a character, the end location of the movement would be a parameter of the action.

Conditions use standard Boolean operators and are based on the values of variables. Several logical terms can be combined to form more complex conditions.

#### 4.2.2 *Stimulus*

Each action set is executed in response to an event in the game world. This event, like actions, is intended to be of a high-level nature. Examples include the event in which a player has finished (successfully or unsuccessfully) a task, clicked a button or waited long enough for a timer to expire. Conversely, events that should not be modeled here are those at the level of raw input signals that are handled in a game engine for player movement, camera control or initiating sequences such as an animation playing. This is in accordance with the overall design goal of this concept of separating the game authoring process from the task of game programming.

Events as those described are referred to as “stimuli” here. An action set may be assigned to each stimulus, indicating that the action set should be executed whenever the stimulus is triggered in the game. Stimuli can be attached to objects, indicating that the associated event has an immediate connection with the object. This could be the “expired”-stimulus of a timer object or the “clicked”-stimulus of a button. Another important stimulus is assigned to each scene, to be triggered whenever the player has entered the scene. A conceptual design choice at this point is to execute the action set each time instead of only the first time a scene is visited. In practice, the choice of triggering it each time allows scenes that are visited more than once

(e.g. a central navigation screen leading to different parts of the game) can have configuration logic that adapts the scene whenever a new visit takes place.

Figure 19 shows an example action set consisting of a stimulus, three actions and a condition determining which of the two bottommost actions is executed at runtime.

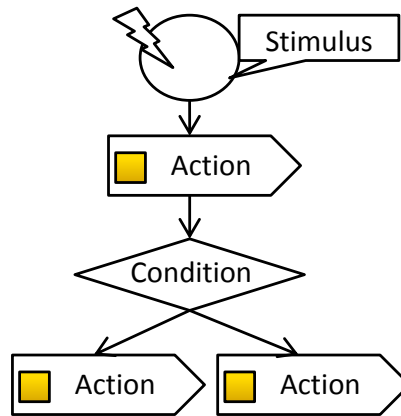


Figure 19: An example action set.

Parallel execution of actions is not explicitly modeled here. This is by design in order to shield authors from the complexities of parallel programs. The only necessary concession to parallel execution is the following: Actions can further be subdivided into those that are considered as requiring execution to halt while they are being executed in the game and those that can be started, allowing execution to continue immediately. We refer to this property as “blocking”, meaning that the software component executing the action(s) will halt execution until the action is finished. This property is included since it lowers the cohesion between the game software (realized in a game engine) and the game model interpreter. As an example, speech acts which trigger virtual characters to speak a sentence will usually be blocking, otherwise, in a sequence of multiple sentences, all the sentences would be triggered at the same time, overlapping each other.

This choice of graph structure for action sets has not been motivated an analysis of the available literature on visual programming languages. A major decision in the matter of graph structure was the choice between a tree structure featuring only branches or a series-parallel digraph, in which each branch of a tree will join the parallel branches again. In effect, this results in a structure similar to a flow network or an electrical circuit with a source and a drain. In terms of game logic, this allows constructs such as adapting a screen text to give personalized feedback based on the performance of a player but then continuing the game regardless of the feedback, e.g. by switching to the subsequent scene. However, the resulting method of authoring might be confusing for authors and therefore is not included in action sets.

ActionSets are evaluated using a depth-first search traversal. Traversal down a sequence is only continued if the associated condition is evaluated to be true. This allows several branches to be executed sequentially, therefore not resulting in a true fork of execution but rather a sequence of conditions and actions. It is left free to the author by specifying the corresponding conditions to create true IF ELSE or SWITCH constructions by specifying completely disjunct logical conditions or to create the equivalence of sequences of IF statements. The latter can be useful in a scene with a central navigational function such as a map. During each traversal of the scene by

the player, the locations that have already been visited have to be removed to keep the player from visiting already visited places.

Since the model for objects and properties clearly defines the parameter space for objects, we can define actions to change each property of an object automatically based on the possible parameters. This can be useful for changing the text or images shown on-screen based on a user input or, in conjunction with timers, even for animating properties.

### 4.3 RUNTIME MODEL EXECUTION

A component for executing the model at runtime and managing the game state based on it is required. Opposed to model-driven development where the model is often translated into program source code and compiled, we suggest that an interpreter is well suited for running games encoded in the model presented here. While translating the model into program code and executing it natively would have the advantage of fast execution, it would come at the cost of having to provide a model translator. Furthermore, the resulting programs would not be easily changeable. Since early tests showed that the processing power required for managing the game model at runtime are neglectable compared to other factors such as graphics, the choice of providing an interpreter has been made.

Due to the story-driven character of games that can be modeled and the already existing term “game engine”, we use the term “story engine” for this component. Figure 20 shows the relationship between an authoring tool, a game engine and the story engine. The story engine is closely combined with the game engine, exchanging information about events in the game and issuing commands to the game. Therefore, it is concerned with the high-level control of the game as well as the adaptation of the game.

The game engine, on the other hand, is responsible for low-level control, i.e. accepting the commands by the story engine and translating them into actual game events, for example by starting an animation or playing a sound. It is therefore responsible for managing the content authors have aggregated using the authoring tool and it supplies the implementation of actual game mechanics.

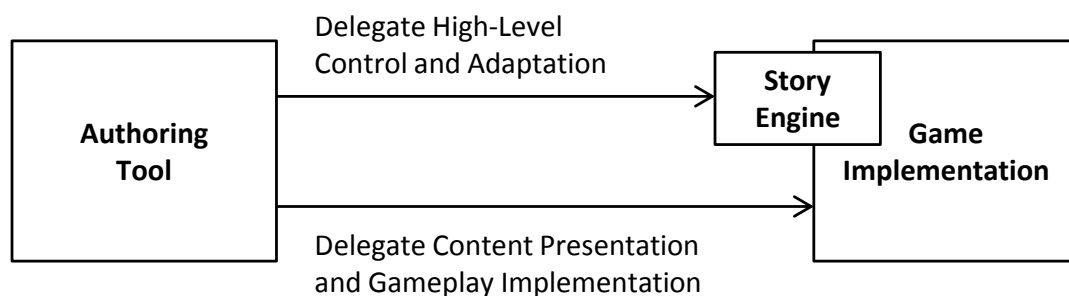


Figure 20: The relation between authoring tool, game engine and story engine.

This architectural pattern is comparable to that of a drama manager described in the context of interactive storytelling and games in chapter 2.

#### 4.4 CONCLUSION

The goal of this chapter has been the provision of a game description model educational games that serves as the foundation for the authoring tool. The two necessary fields to be modeled have been identified as the game's structure and logic.

For the first model, a discussion of the related models from interactive storytelling, multimedia authoring and game development yielded in the choice of a storytelling-based model in which games are broken down into scenes. We elaborated this model and defined elements that are sufficient to describe all relevant structural aspects of a game. This includes the hierarchically organized scenes for managing minimal units of games, transitions for connecting scenes with each other, objects for modeling game objects contained in scenes as well as parameter for scenes and objects to describe instances of these types further.

Based on the examination of visual programming languages, we presented a model of game logic that allows authors to work on a high level of abstraction. The basic unit of this aspect of the game model is an action which specifies a high-level command issued to an object. Branching sequences of actions are realized as action sets. The branching is realized using conditions which indicate the parts of an action set which are conditionally executed at runtime. Action sets are triggered by stimuli, which are high-level events from the game.

By the provision of this game model, authors in an authoring tool can work on the structure and logic of the game without using a programming language. The model is executed in a story engine, an interpreting command unit which introduces a level of abstraction between game engines and the authoring tool.

In the following chapter, the concept is extended towards adaptive games by introducing adaptive algorithms that the author can yield some control to in order to allow a game to be adapted for a player at runtime.

IN this chapter, the concept assembled in the previous chapter is applied to the creation of adaptive games. Before the details of the extensions for adaptivity are shown, we turn towards an examination of mechanisms for adaptive games and the benefits of an authoring tool supporting them directly.

We use the term “adaptive game” to refer to a game that can be adapted to a certain player. Note the separation between “adaptive” and “personalized” used in the following text: we use the former to refer to the game adapting to the player during play, while our use of the second refers to a process in which a (possibly persistent) profile for a certain player is created and updated continuously over several play sessions. Adaptation and personalization can be sorted according to two criteria:

- *Initiator of adaptation/personalization*: The entity initiating the adaptation could either be a human or artificial (e.g. a machine learning algorithm).
- *Time of adaptation/personalization*: The adaptation could happen before play (resulting in a pre-configured game) or during play.

When we assemble these axes into a matrix, the variations of adaptation and personalization shown in table 5 become apparent (combinations, e.g. a preconfigured version being refined during runtime, are not shown here).

Of the four possible adaptation schemes shown, we pursue the two schemes “Manually customized games” and “Player modeling” in the following. The scheme of automatically created customized versions of a game is not followed here since it is not immediately connected with an authoring tool used by human authors. The other scheme not further examined can be identified with the term “Game Mastering”, which originated from pen and paper role-playing games and refers to a person participating in a game who has deeper insight into the game world than the other players (e.g. by knowing parts of the game world yet unknown to the players) and who controls parts of the game. In a digital game, a Game Master can have detailed information about the performance of players and adapt the game world to promote

|             | HUMAN  | ARTIFICIAL   |
|-------------|--|--|
| BEFORE PLAY | Trainer/Teacher creating a customized version of a game for a target audience or one specific player | Creating customized versions of a game by artificial intelligence or machine learning. |
| DURING PLAY | Game Mastering   | Player Modeling  |

Table 5: Possible adaptation schemes based on the initiator of the adaptation and the time of adaptation.



the purpose the players are working on in the game as demonstrated by Wendel et al. [156]. Necessary research in this area focuses on adaptation interfaces for the game master which provide relevant information and allow the game master to influence the game. Since this process is again not directly connected to the authoring process, we will not focus on this process in the following.

The adaptation scheme of manually customized versions of a game is directly suited for situations in which a clear set of target audiences with different properties (e.g. varying background level, time constraints, etc.) are addressed and in situations in which there is a one-to-one relation between author and player. The first possibility is seen in situations such as an interactive tour guide for a museum realized as a game. The author could make one very short version for players with very little time, a medium length game and a long game for players interested in many details or with a lot of time available. The second possibility is seen in trainer/trainee or therapist/patient<sup>1</sup> relationships, where a domain expert is responsible for a single player. An example is a teacher customizing an educational game according to the current curriculum and the previous knowledge of the class.

The settings the author makes can in this case be personalized for the specific player, based on the expertise of the author. For example, in an exergame, the author could be a therapist hand-tuning values such as the length of the game, the intensity to be reached (concerning heart rate or other vital parameters) to be maximally effective for one patient. This is especially relevant since exergames often require special input devices and sensors (for effort measurement, movement data, ...) that are connected to the game. For example, in a game where a bicycle ergometer is used as the input device, parameters such as energy, pedal frequency, velocity and resistance are included in the game. In the authoring tool, the influence of these sensor data on the game in relation to a target goal for the patient has to be configured.

By utilizing the game models in the authoring tool, the creation of variations of games such as described here is easily accomplished. Authors can for example copy & paste scenes in order to create a variation with different parameters, or use simple conditions to switch between different paths through the game based on a variable such as the difficulty or time to play through the game.

Authors who fine-tune a game to a specific player are supported by the concept of domain-specific editing as is detailed in the following chapter. In a first step, an experienced author creates a basic game including all required interaction templates and objects and the necessary game logic. Later, a less experienced domain expert can switch to a domain-specific authoring mode and simply work on the relevant parameters for the intended player without having to change or work with other parts of the game.

In Player Modeling, the state of the game is monitored during play and a computer system uses the information about the game state to adapt the game based on certain objectives.

In order to carry out modeling, we need to determine which aspects of the human player we wish to model. We argue that the following areas are critical for an educational game:

- *Storytelling*. Interactive narratives are an effective means for engaging players and embedding knowledge. The goal of modeling narrative lies in upholding

<sup>1</sup> See appendix A.10 for an example of an exergame with this customization scheme.

the suspense, immersion and emotions of the player by tracking the state of the narrative.

- *Gaming*. The key aspect of educational games is that they embed the serious purpose into a fun game. Therefore, modeling concerning the game should be used to adaptively enhance the enjoyment and motivation of the player.
- *Learning*. In most areas of educational games, learning in some way is involved. Therefore, tracking the learning success and motivating players to learn is an important area of modeling which is therefore included in this concept.

While these areas mostly complement each other (for example, the combination of storytelling and gaming is often found in games such as adventure or role-playing games) and can increase the motivation and effectiveness of a game, adaptation based on these areas is faced with several problems due to the combination of goals. The “narrative paradox” introduced in section 2.2.2 refers to the problem that authors of a narrative wish to effect as much control as possible over the game in order to control the narrative arc and be able to control the story at all times. However, a player in a game expects interactivity and a game world which reacts to input, thereby conflicting with the interests of the author. The story has to be created in such a way that it can evolve from the input of the player, leading to emergent narratives ([82, 87]).

Similarly, gaming and learning should be well integrated and not be realized as a series of tasks where gaming elements and learning elements alternate. Instead, gaming situations and learning situations should be merged, resulting in scenes that advance the narrative, include motivating play elements and covering learning content simultaneously.

In the following, we first describe how the individual aspects of storytelling, gaming and learning are to be modeled, and derive a combined model referred to as Narrative Game-Based Learning Object (NGLOB). Following this definition, we describe three steps that form the basis of the adaptation loop that will happen during gameplay. In the first step, an action by the player is detected and interpreted to indicate updates to a model. These model updates are then carried out to result in a new state of the player model. Finally, whenever adaptive changes to the game are to be done, the model is used to determine which change is optimal in the current situation. The adaptation cycle that results from this is visualized in figure 21.

Note that we concentrate on a macro-adaptive approaches here (see section 2.2 for the definition of micro- and macro-adaptive approaches). Micro-adaptive interventions can largely be realized within the educational game model described so far since authors mainly need local information from one specific scene to author them. Conversely, macro-adaptive approaches require global knowledge about the current user model and need to be aligned with the possible game paths through the game. Since this information is not captured in the game model so far, we focus on extensions for this information here.

Finally, we describe the integration of this adaptation loop in the game model and the authoring process.

For each of the relevant aspects of adaptive games, a choice of a modeling approach has to be chosen. In the following, the individual aspects are examined and a

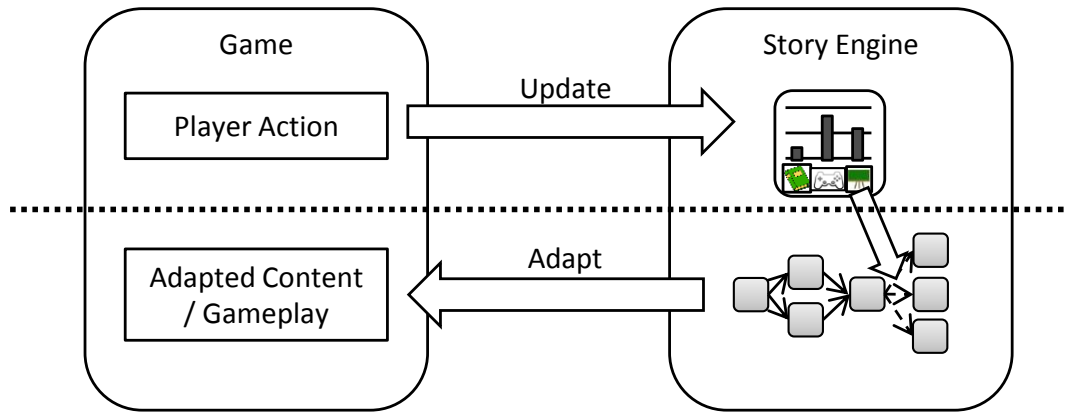


Figure 21: The cycle of updates and adaptations.

combined approach is presented. Since the analysis and choice of appropriate models is immediately based on the properties of the models, they are presented in detail here instead of chapter 2.

### 5.1 NARRATIVE MODEL

The purpose of a narrative model in the context of adaption is to structure the game in such a way that the resulting narrative is of high quality, i.e. by supplying a dramatic arc, while simultaneously allowing for alternative paths through the game. Narrative models structure a story into typical stages, complete with archetypical roles and tasks that characters assume or work towards during the respective stage.

In order to model narrative structures, researchers and developers turn towards the available literature from narratology. Common examples of models that have been created to model classical, non-interactive stories include the following by Propp and Vogler respectively Campbell.

Propp [115] described a model of stories that was used to describe Russian folk tales. It is composed of 31 functions that can appear in sequence in a story. Furthermore, the model includes 8 character types that are common in stories. An example for a set of story functions that are found in the center of many stories are such functions as “departure”, in which the protagonist leaves his or her home; “struggle”, which characterizes the fight with an adversary, followed later by “victory” and “return” to the old world of the hero.

The Hero’s Journey is a model derived by Joseph Campbell [22] from an analysis of classical stories and myths. He identified a set of common stages many popular stories share. The central element of this model is the development of the narrative’s protagonist or hero. The model is constituted of a series of stages, in which a hero starts out in his or her “ordinary world”, visits a “special world” in which an adventure is undertaken and returns, changed in some way, to the ordinary world. Campbell’s model consisted of 17 stages. Vogler [155] later analyzed the application of the Hero’s Journey to contemporary narratives, for example in screenplays for Hollywood movies. His version of the journey, the “Writer’s Journey”, consists of only 12 stages that follow the same overall structure as Campbell’s. Figure 22 shows

a schematic diagram of an interactive narrative structured using the stages of the Writer's Journey.

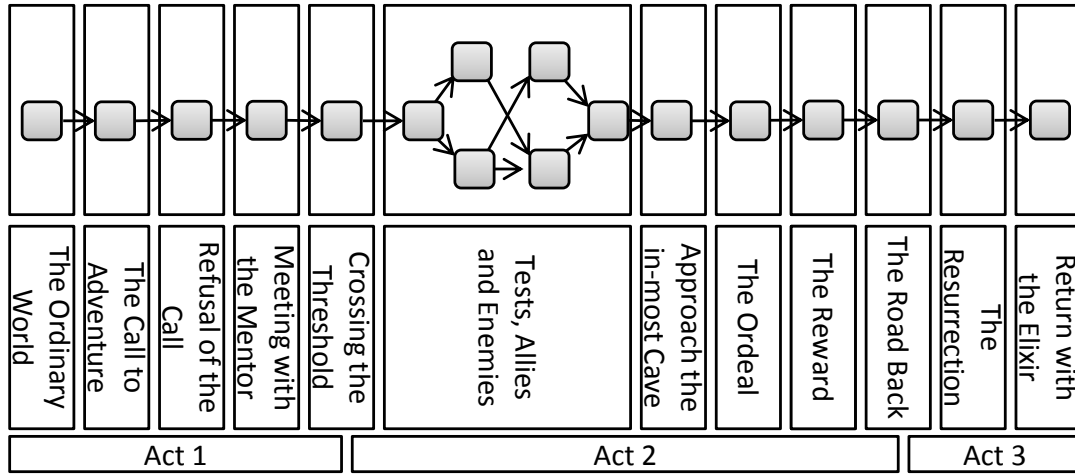


Figure 22: The Writer's Journey story model applied to a game.

Since the Hero's respectively Writer's Journey story model is well applicable to adaptive storytelling and has been used successfully in educational games (especially story-driven adventure games), we have chosen the Writer's Journey as the most appropriate existing story model to achieve a good trade-off between adaptation and free gameplay on the one hand and suspenseful narration on the other hand.

## 5.2 PLAYER MODEL

The goal of player modeling in this concept is to adapt games in such a way that the players' motivation to play is increased and thereby the purpose of the educational game is achieved efficiently. However, it has to be noted that the models chosen here have to be, by definition, only broad generalizations of human preferences and learning behaviors, due to the facts that they are required to be machine-interpretable and manageable by human authors who have to supply annotations for the models.

The goal of modeling players can be achieved using two approaches that differ in the set of properties of players that are to be observed to form the model. The first approach is a general, personality-oriented approach, in which the target is a psychological personality profile of the player. The second approach, on the other hand, strives towards differentiating players into a set of abstract player types, and aims at determining how the highest interest in the player can be raised.

### 5.2.1 Personality-Oriented Models

When analyzing the first, psychology-oriented approach, we find psychological models such as the Myers-Briggs Type Indicator (MBTI) [17], based on Carl Gustav Jung's psychological types. This and other, more refined systems based on it, are commonly used in corporate environments in areas such as customer relationship management, conflict resolution or strategic composition of teams and others.

The MBTI differentiates four human personality areas with two exclusive characteristics. The four areas and characteristics are defined as shown in table 6.

| AREA:                   | CHARACTERISTIC 1: | CHARACTERISTIC 2: |
|-------------------------|-------------------|-------------------|
| ATTITUDE TOWARDS OTHERS | Extraversion (E)  | Introversion (I)  |
| INFORMATION ACQUISITION | Sensing (S)       | iNtuition (N)     |
| DECISION PROCESS        | Thinking (T)      | Feeling (F)       |
| LIFESTYLE               | Judging (J)       | Perceiving (P)    |

Table 6: The Myers-Briggs type indicator, comprised of four personality areas and two exclusive characteristics.

The type indicator is used to classify the person by listing the characteristics in each area, for example a person whose characteristics are Introversion, iNtuition, Thinking and Perceiving is classified as INTP. Therefore, the model allows 16 different personality types. Based on this list, personality aspects can be inferred.

The MBTI was later extended by Keirsey and Bates [66], resulting in the Keirsey Temperament Sorter (KTS). This extension further uses sub-temperaments.

For the MBTI and KTS, official question catalogues to be used in classification have been published. In an implementation of player modeling for educational games, these catalogues can be used as guidelines for good assessment tasks that can be interpreted and used to update the player model.

### 5.2.2 Player-type based Models

The second category of models focus on the types of game that different players enjoy and is therefore more aligned with games than the personality-approaches outlined above.

Bateman and Boon [9] present a categorization that is rational and economically inspired. They define the following four types, which are based on players' play and buying preferences:

- Conqueror: Players enjoy being dominant within the game or in the social circles (e.g. forums) around the game. They wish to win at any cost and enjoy competition.
- Manager: Process-oriented players who focus on logistics and in-depth understanding of the game in order to master it. Players play repeatedly in order to train themselves and learn more about the details of the game.
- Wanderer: Players continually searching for new enjoyment in the game, less competition-focused than other types.
- Participant: Enjoy playing in groups and the aspect of escapism by immersing themselves in the game world.

The Gaming-Narrativist Theory (GNS) theory is the result of theoretical analysis of pen and paper role-playing games, carried out by Ron Edwards<sup>2</sup>. This model is relevant since role-playing games centrally feature a shared control over the game

<sup>2</sup> <http://www.indie-rpgs.com/articles/3>, last visited on January 24, 2013

by a game master adapting the game to the actions of the players, while following certain narrative or ludic goals.

The analysis showed three non-exclusive types of players:

- **Gamist:** This type of player cherishes competition and is interested to learn the conditions for winning and losing. The game is seen as an arena for competitive play.
- **Narrativist:** The narrativist focuses on the creation of a quality story, including good protagonists and an overall theme of the narrative. The game should be about an ethical/moral conflict and is seen as providing the material for constructing the narrative.
- **Simulationist:** A simulationist sets out to maintain consistency in the play-experience and the logic of the game world. He or she thereby tries to enhance one of the five elements of RPGs that Edwards identified: Character, System, Setting, Situation and Color .

The next model we present is that by Laws [83], who defines, again in the context of pen & paper role-playing games, the following six player types:

- **Power Gamer:** Strives towards improvement of aspects of their character (e.g. concerning strength, size or wealth). Power Gamers adhere to the game rules exactly, seeking to exploit those that maximize their gain with minimal effort.
- **Butt-Kicker:** This player intends to beat enemies and overcome challenges, in order to vent anger. Adheres to the game rules in order to maximize his or her character's strength and readiness for battle.
- **Tactician:** Prefers logical and realistic obstacles that can be overcome with complex, militaristic solutions over role-playing and acting in correspondence with the traits of characters when it leads to tactically unsound behavior.
- **Specialist:** Always chooses one type of character and demands rules and situations that apply to this character.
- **Method Actor:** Wishes to identify strongly with their character and makes decisions in the game based on their understanding of the character's psychology, even if choices are in conflict with rules or the higher good of the group.
- **Storyteller:** Values the overall story told in the game over the game system or their character's personality and seeks to continue the narrative.

Laws further describes casual gamers who do not have any deeper interest in the game and therefore don't have real preferences which can be used in player modeling.

The final model described here is that developed by Bartle [8] after researching and developing Multi-User Dungeons (MUDs), precursors of the later popular Massively-Multiplayer Role-Playing Games (MMORPGs). MUDs are akin to interactive fiction, i.e. completely text-based adventure games in which several players could participate.



The success of MUDs before the increase of graphical multi-user games led Bartle to inquire about the motives of players playing MUDs. Bartle identified four player types and the associated mindset players of this type have:

- Achievers: Construct game-like goals for themselves, such as amassing treasures or beating enemies, and work towards them.
- Explorer: Wish to find out the details of the simulated world, starting with the spatial configuration of the world, followed by the higher levels of game rules and simulation parameters.
- Socializer: Enjoy communication with other players via their in-game avatars.
- Killers: Seek conflict in the game and use the game to hurt others.

Similar to other models, these types are non-exclusive and players might identify with several types. Bartle however argued that most players prefer one of the four types.

After examining four possible models for player modeling, we discuss their properties and choose the most applicable model for our concept.

GNS, the first presented player-type oriented-model, is specific to role-playing games, not only referring to the characters in the game, but also to external factors such as the role-playing game system itself. As a model to be used in adaptive educational games in general, it is therefore not suitable. However, the properties of the player types (focus on competition, realism or suspenseful narratives) are worthwhile to keep in mind in the design of adaptive game features.

The model by Laws, created originally for pen & paper role-playing games, can also be applied to digital games especially those based on a narrative (which applies to the concept described here) and the development of characters. As noted above, Thue et al. [144] use it in their PaSSAGE system.

Since the model by Bartle was created especially for digital (adventure) games (MUDs), it is by definition applicable to adaptive digital games. It has previously been used for digital games, for example in the study of 35.000 players of Massively Multiplayer Online Games conducted by Yee [160] where it was used as the basis of a questionnaire. Furthermore the low complexity of the model is advantageous. Authors who use the model have to understand the player types and choose the appropriate categorization. This implies that the number of categories should be minimized while staying useful, since authors will have to annotate many game scenes with them, a process that could be hindered by an overwhelming number of categories.

Due to the applicability and low complexity as well as the clear definitions of the player types, the model by Bartle has been chosen as the default model for the player-modeling aspect of adaptivity. However, any similar model sorting players into categories is applicable.

After motivating our choice of Bartle's player-type oriented approach for a player model, we now present how this model is encoded to be used in an adaptive game. Two methods for encoding player models used in games are presented, the method by Houlette and that used in PaSSAGE.

Houlette [59] originally published his approach in the context of artificial intelligence for NPCs. It is based on a vector of player attributes (here referred to as

“traits”) that relate to the game. Houlette uses the example of a role-playing game in which a player can use magic to defend himself or others. This could be encoded as a trait “usesDefensiveMagic”, based on which the game can adapt the behavior of other characters in the game. For example, the player’s allies could be more offensive since the player himself enjoys defensive play. For updating the traits, the LMS-algorithm [158] is used, due to its robustness, low complexity and speed.

The PaSSAGe approach by Thue et al. [144] was created for interactive storytelling. Similar to the previous encoding, it is based on a vector of “player preferences”, which defaults to a subset of the model by Laws (cf. previous section). The approach assumes a set of story events that players can experience in a freely explorable world. These events are annotated with a vector of attribute values and describe the appropriateness for a given player type. By comparing the vector with the player’s vector, the appropriate events for a player are chosen. The vectors can be of any dimension. Thue et al. propose the range  $[-4, 4]$ . In order to build the vector of the player, actions by the player are annotated with another vector. The model is updated using this vector whenever the action is carried out by the player.

The next appropriate event is chosen using an event suitability function, whose formulation is (cf. [144, p. 640]).

$$\text{eventSuitability}(K) = \sum_{i=1}^n (a_i \cdot \max(x_i, 0)) \quad (1)$$

$K$  refers to the event respectively the story node,  $a = (a_1, \dots, a_n)^T$  is the author annotation vector and  $x = (x_1, \dots, x_n)^T$  the vector of the player model.

An entry  $a_i = 0$  indicates that an event should not be taken into consideration,  $a_i < 0$  indicates an aversion and  $a_i > 0$  an affinity. In the player model, negative values are clamped to 0, motivated by an observed increase in performance due to Thue et al. [144].

The optimal event is that with the highest suitability  $K_{\text{best}}$  with

$$\text{eventSuitability}(K_{\text{best}}) = \max(\text{eventSuitability}(K)) \quad (2)$$

where  $K = \{K_1, \dots, K_n\}$  represents the set of all possible events.

### 5.3 LEARNING MODEL

The task of a learning model in adaptive games is to ensure the learning success of the player. We can concern ourselves with the learner type of the player or with the optimal learning path for the player’s current knowledge.

Concerning learner types, we find several models that have been proposed. Kolb’s [80] model is based on two axes, the first one describing the way learners perceive the learning content, with the two extremes of “concrete experience” and “abstract conceptualization”. The second axis is that of transformation of perceptions into knowledge, described by the extremes of “reflective observation” and “active experimentation”. By categorizing learner types along these axes, Kolb [80] arrives at the following learner types:



- **Converger.** Enjoys application of theories in practice and learns well when the process of learning has clear goals.
- **Diverger.** Creative learners who are able to see situations from new perspectives.
- **Assimilator.** Theory-driven learner who enjoys constructing a theoretical model that is considered more important than the practical application.
- **Accommodator.** Practical learners who apply trial-and-error to solving problems and improvise well.

A number of other learner type taxonomies exists. A meta-study was carried out by Gregory [51] that resulted in the following learner types “Puppy” (similar to Kolb’s “Diverger”), “Microscope” (similar to Kolb’s “Assimilator”), “Clipboard” (similar to Kolb’s “Converger”) and “Beach Ball” (similar to Kolb’s “Accommodator”).

While it could be reasoned that modeling an educational game to be adaptive towards learning types is beneficial, we believe that digital games can inherently cater to the different learner types by offering information in various ways (visually, acoustically, explorative; cf. [109]). The learning aspect of adaptivity should focus on keeping the game motivating and keeping the learner in the flow zone of optimal learning, a task that is not achieved by adapting to learner types per se.

For a useful model for encoding the learning state of a player, we turn towards the behavioristic Knowledge Space Theory (KST) [37] and the later cognitivist extension to CbKST [2]. In CbKST, the learning domain is structured as a set of problems in a prerequisite relationship, which is extended by the prerequisite competencies for the problems to form the competency-based knowledge space.

The knowledge space is a directed graph composed of a set of problems ( $Q$ ) and a surmise relationship between the problems. This transitive, reflexive, antisymmetrical and binary relation  $q \Leftarrow t$  indicates that we can infer heuristically that problem  $q$  can be solved correctly by a learner when he or she can solve problem  $t$ . This also indicates that the competencies required in solving problem  $q$  are prerequisite for solving problem  $t$ . As an example, we can examine the knowledge of where Berlin is located on a map. One possible assumption is that someone who can locate Berlin on a map also needs to know the location of Germany on a map. Being aware of such relationships can help in educational settings in many ways, for example by reducing the number of irrelevant or redundant hints. For example, if a learner has demonstrated that he or she can find Berlin on a map, they probably do not require additional learning content that teaches them how to locate Germany on a map. Figure 23a shows a knowledge structure for the set of skills  $\{a, b, c, d\}$ .

In the 80Days project for adaptive games, a story structure was combined with CbKST, resulting in a combined game graph consisting of annotated story situations with information how players can move along a useful order of prerequisite skills [72]. Figure 23 shows example structures for story, learning and game graphs. The diagram in Figure 23b shows different ways in which the set of skills  $\{a, b, c, d\}$  can be learned in an order that reflects the prerequisite relationship between them. Figure 23d shows one possible result of combining the story situations in figure 23c with the learning paths. Note that some scenes allow for adaptation based on varying the narrative (e.g.  $(F, \{a, b, c, d\})$  and  $(G, \{a, b, c, d\})$ ) which contain the same learning

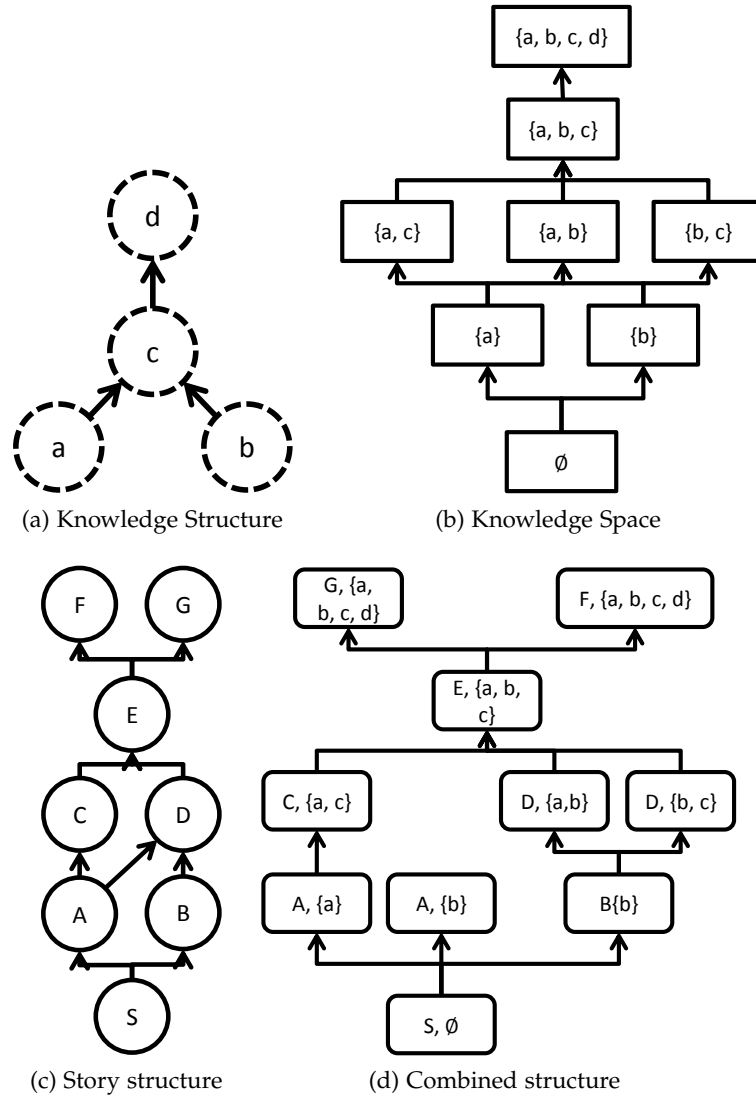


Figure 23: The combination of a knowledge space and a story structure along which the competencies  $\{a, b, c, d\}$  can be taught.

content) or on varying the educational content (e.g.  $(A, \{a\})$  and  $(A, \{b\})$ , which teach different educational content with the same narrative content). This could be realized for example by a scene in which the player has a dialogue with a character, and only the bits of the dialogue relating to the educational content of the scene are changed.

Due to the positive results achieved by introducing CbKST-based learner modeling in the ELEKTRA and 80Days projects, we use this method for learner modeling in the NGLOB concept.

#### 5.4 NARRATIVE GAME-BASED LEARNING OBJECTS

After describing the three most relevant adaptation criteria for educational games, we combine these axes in the form of Narrative Game-Based Learning Objects (NGLOB). This concept was first published by Göbel et al. [49].

A NGLOB is derived from a scene in the game model by adding annotations for each of the aspects concerning the appropriate context of the scene.

We encode each aspect of a NGLOB using the respective context. In the following, the machine-readable part of the annotation is described. The definition for an NGLOB also includes a textual description of each context, which is intended to help authors capture their ideas for the NGLOB and document it.

Concerning narration, we make use of story models that can structure a story into stages that describe the purpose a scene has in the overall narrative. While several models are possible, we suggest the Writer's Journey as reasoned for above and provide this as the default story model.

The narrative context  $C_N$  is concerned with the appropriateness of a scene in a given story model. It is encoded as a list of 2-tuples (storymodelStep, appropriatenessFactor) with the storymodelStep referencing a step in a story model and the appropriatenessFactor, ranging from  $[0, 1]$  indicating the appropriateness.

For adaptation to play preferences, we propose to utilize a player model that can capture the different types of gamers and their preferences. The concept of NGLOBs is flexible concerning the specific player model, as long as it differentiates into different classes. While we use the model by Bartle as described above as the default model due to its simple structure and common usage in games, other models can be used or the chosen model can be extended by authors by adding new criteria specific to the game genre or the specific game's purpose. For example, game-specific traits like those used by Houlette [59] could be added to a more general player model.

The gaming context  $C_G$  is therefore concerned with the appropriateness of a scene for the different player types. It is realized as a list of 2-tuples (playerAttribute, appropriatenessFactor). A playerAttribute references a type of player (the default is the model by Bartle), whereas the appropriatenessFactor indicates the likelihood that this scene appeals to the specified player type. It is, again, in the range  $[0, 1]$ .

Adaptation concerning learning goals is carried out by using a learner model that works on captured according to CbKST. In a first step, an author defines the curriculum of the game as a graph containing competencies and their relationships. NGLOBs are then annotated with the competencies that the object requires in order to be understood.

This learning context  $C_L$  is realized as a vector with two lists. The first entry lists all the skills that are associated with the NGLOB, i.e. all skills for which learning content in some way is presented in the scene. The second entry lists all prerequisite skills which are necessary to be able to understand the associated skills in the NGLOB or to be able to solve the task included in the NGLOB. Note that that associated skills are used only for planning purposes, the actual update is carried out based on annotated player-initiated events.

These three contexts are combined into a three-tuple  $C_N \times C_G \times C_L$ , resulting in the complete metadata set of an NGLOB. An example is shown below, with three appropriateness values for the Writer's Journey story model, the appropriateness for the player model by Bartle and 2 associated and prerequisite skills.

$$\left( \left\langle \begin{array}{l} (SM\_HJ\_4, 0.5), \\ (SM\_HJ\_5, 0.8), \\ (SM\_HJ\_6, 0.2) \end{array} \right\rangle, \left\langle \begin{array}{l} (PM\_BA\_K, 0.0), \\ (PM\_BA\_A, 0.2) \\ (PM\_BA\_S, 0.8), \\ (PM\_BA\_E, 0.2) \end{array} \right\rangle, \left( \begin{array}{l} \langle A101, A102 \rangle \\ \langle P201, P242 \rangle \end{array} \right) \right)$$

Apart from augmenting scenes with metadata for adaptation, the game model has to be extended to allow adaptive choices of scenes. While transitions before were triggered explicitly, we now extend them to be either “fixed” (triggered explicitly) or “free” (triggered indirectly). When a scene is connected to several others via free transitions, this indicates that the author indicates that the next most appropriate scene for the current context (as determined by the player model and the scene annotations) should be chosen adaptively. The advantage of this approach is that authors have an overview of all possible paths that the game has and can see which previous points lie on the path of the player, allowing authors to plan the narrative better.

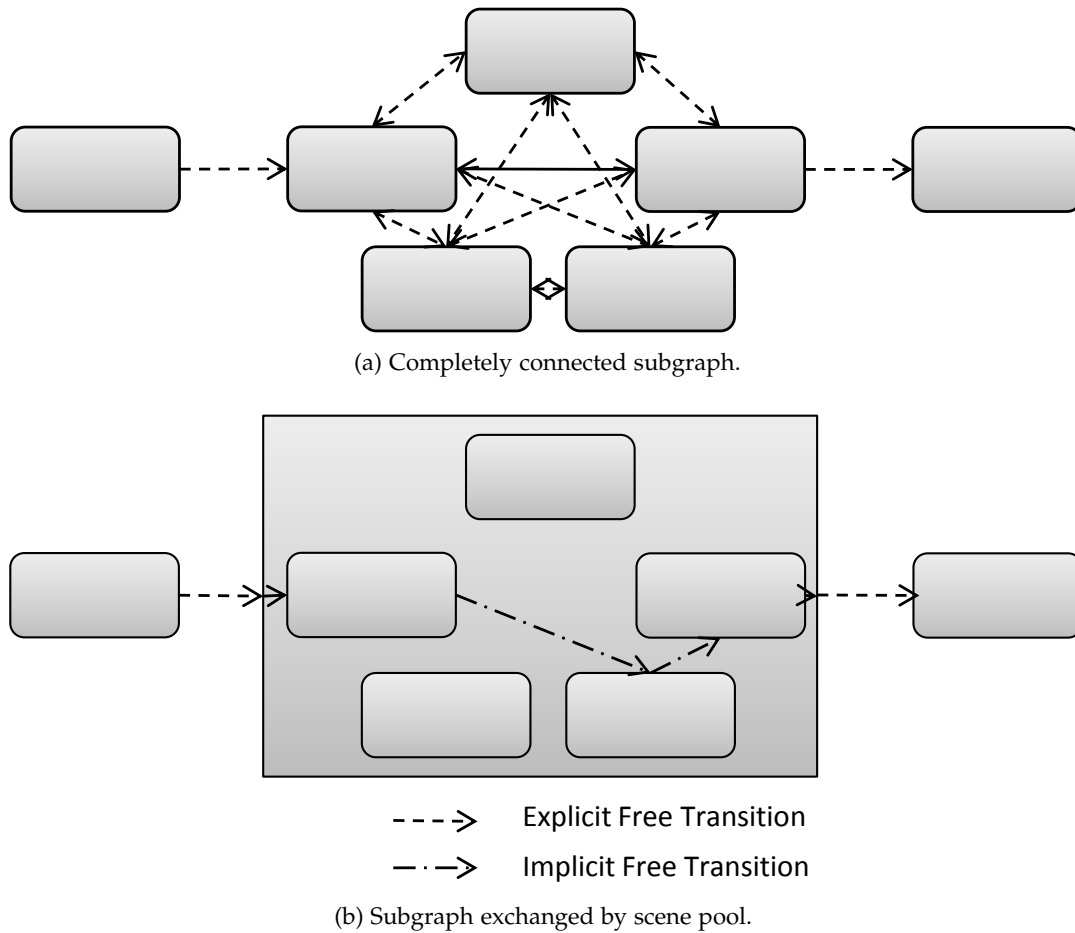
If the number of branches becomes too large or if the game is intended to be more modular, the method of marking transitions as free becomes cumbersome and confusing for authors. The worst case-scenario would be a game with many scenes that would end up being interconnected with all other scenes, a configuration that is hard to manage for an author. Instead, we propose that these scenes can be grouped together in a pool of scenes. Since the game model allows scenes to be nested, we add a new property to scenes that allows them to be marked as pool containers. This indicates that all pairs of scenes in this container are connected via implicit free transitions. During play, when this pool is reached, a number of scenes is visited adaptively before the container is left again. Figure 24 illustrates this concept.

The method of scene pools described here allows modular authoring of games, since the game is essentially assembled at runtime and, due to the large number of adaptive choices, can be fine-tuned to a player. However, it is very abstract for authors since it departs from traditional linear storytelling and does not allow authors to immediately see the possible paths through the game. Storytelling (concerning the order of narrative events), learning (concerning meaningful and good learning paths) and gaming (concerning the difficulty curve of the game) all are impacted, since the sequence of scenes is not pre-determined. Therefore, the use of a rapid prototyping tool as shown in the following chapter becomes imperative in order to find out potential problems in the game. In order to retain an overview, authors can resort to the String of Pearls model (cf. figure 5c on page 26) in which important narrative events are not included in scene pools (the “pearls”) but rather frame the pearls, so that their order and occurrence during each play session is guaranteed.

In the remainder of this chapter, we discuss the algorithms used at runtime to control NGLOBs and the integration into the authoring process. This includes the adaptive cycle as displayed in figure 21 on page 64

#### 5.4.1 Game Event Interpretation

The first step in the cycle of adaptation is the interpretation of game events. In order to use player modeling, events the player initiated or information about the game state (such as elapsed time) have to be interpreted and turned into data points to



(b) Subgraph exchanged by scene pool.

Figure 24: Scene pools.

be used by the model update algorithms in the next step. Keeping in line with the high-level nature of the model described in this thesis for far, we do not concern ourselves with all individual events such as mouse clicks or keystrokes, but rather work at the levels already established with stimuli. Further information that is not related directly to stimuli is the information about the elapsed time between events or since the start of a game or a specific scene.

As an example, finishing a task successfully that requires certain skills implies that the player has mastered these skills. Similarly, choosing a branch in the narrative which is governed by social interaction ( $\rightarrow$  Socializer) or exploration ( $\rightarrow$  Explorer) can give information about the player model. Such events are referred to as game evidence in others systems, see for example Peirce et al. [109] for an example from the ELEKTRA project.

#### 5.4.2 Model Update

Whenever a relevant event has been detected, the respective model is updated. This update should be carried out with a balance between older and current information about the player in mind, in order to lower the effects of errors in measurement (i.e. the game evidence was misinterpreted) and those due to concept drift (i.e. player preferences and knowledge changing during play, cf. [153])

This problem can be seen as an instance of an optimization problem and be solved with computer science methods in this area. Bellotti et al. [10] provide an architecture for adaptive serious games that proposes to solve this problem using genetic programming.

We propose to use the Least Mean Squares algorithm as introduced by Widrow and Hoff [158], which is commonly used in machine learning, neurological networks or adaptive filters. This algorithm learns tendencies in the input data by combining a known value and a new value with a weighting factor. The resulting value serves as the known value in the next iteration of the algorithm.

We refer to the new value as  $v_{\text{new}}$  which is combined with the known value  $v_{\text{old}}$  by means of a weight  $\alpha$ , by the formula

$$v_{\text{new}} = v_{\text{old}} \cdot \alpha + (1 - \alpha) \cdot v_{\text{observed}} \quad (3)$$

$\alpha$  is in the interval  $[0, 1]$ , therefore, the formula results in a linear combination of the old and new values and determines the influence of new values on the measurement. New values will, over time, influence the overall value of a tracked property and therefore adapt to the target model, while not adapting to short-term changes or instabilities too much.

An alternative to the LMS algorithm was introduced in the PaSSSAGE system. Here, the new value and old value are combined by addition:

$$v_{\text{new}} = v_{\text{old}} + v_{\text{observed}} \quad (4)$$

We see that the values of  $v$  are potentially unbounded.

The two update algorithms were compared in an evaluation in which a human player with a certain player preferences vector was simulated to play a randomly generated game which included game events annotated for each player type. Hereby, a game structure was created at random that included both free transitions and annotated actions. A virtual player with a fixed player model consisting of four attributes (similar to Bartle's model) was simulated, that always acted in such a way that was maximally consistent with the player model. Simultaneously, the LMS and PaSSAGE player modeling approaches were tasked with learning this player model. The relative error of the two approaches is shown in figures 25 and 26. The study was carried out under supervision in the context of the thesis prepared by Lukas Pruschke (see appendix D for supervised theses). The results have been confirmed and are reproduced here.

While the player model remained constant in the previous simulations, in a second simulation, the player model was changed in the middle of the second run to simulate concept drift. This resulted in the play traces of LMS (figures 27 and 28) and PaSSAGE (figures 29 and 30).

The tests showed that the PaSSAGE approach, when used with a fixed model, showed better performance when considering the relative and overall error rate of adaptive choices. However, if the simulated target player model changed over time (emulating a player who acts inconsistently), the flexibility of the LMS algorithm performed better. Considering the assumption of Bartle that players usually identify strongly with one of the player types and therefore do not act inconsistently during

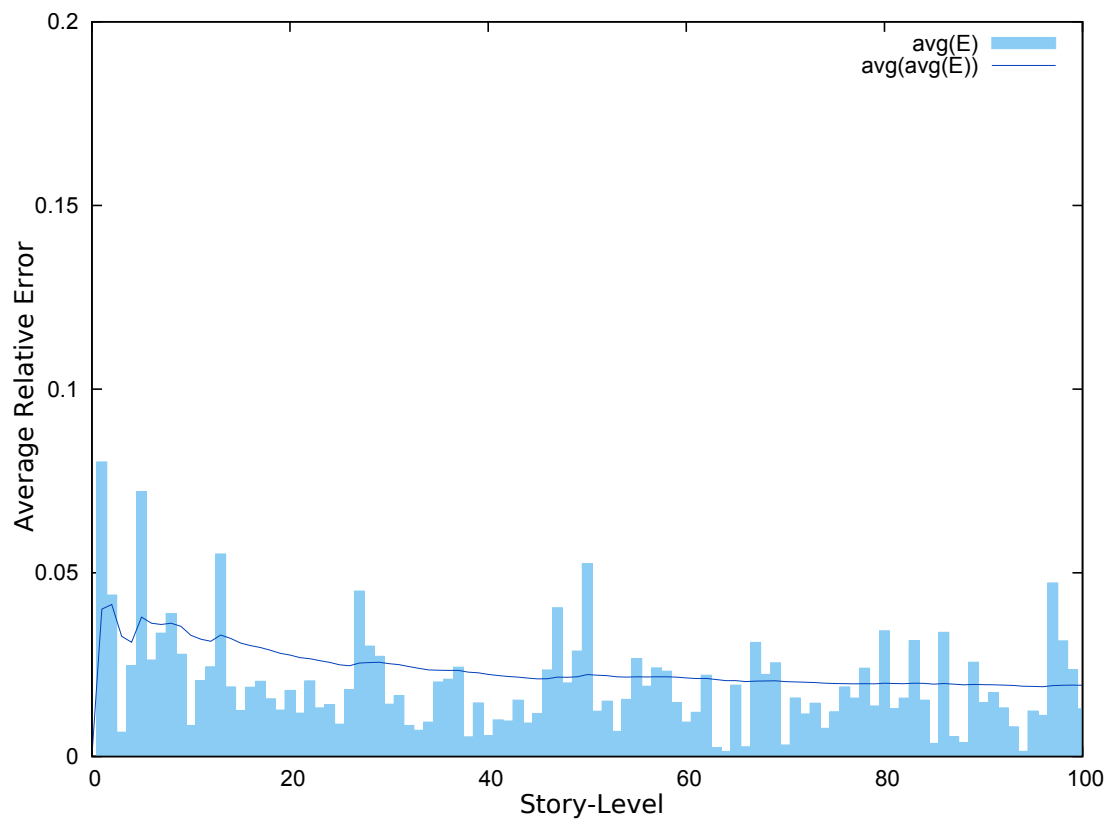


Figure 25: LMS: Relative error (average of 50 generated games).

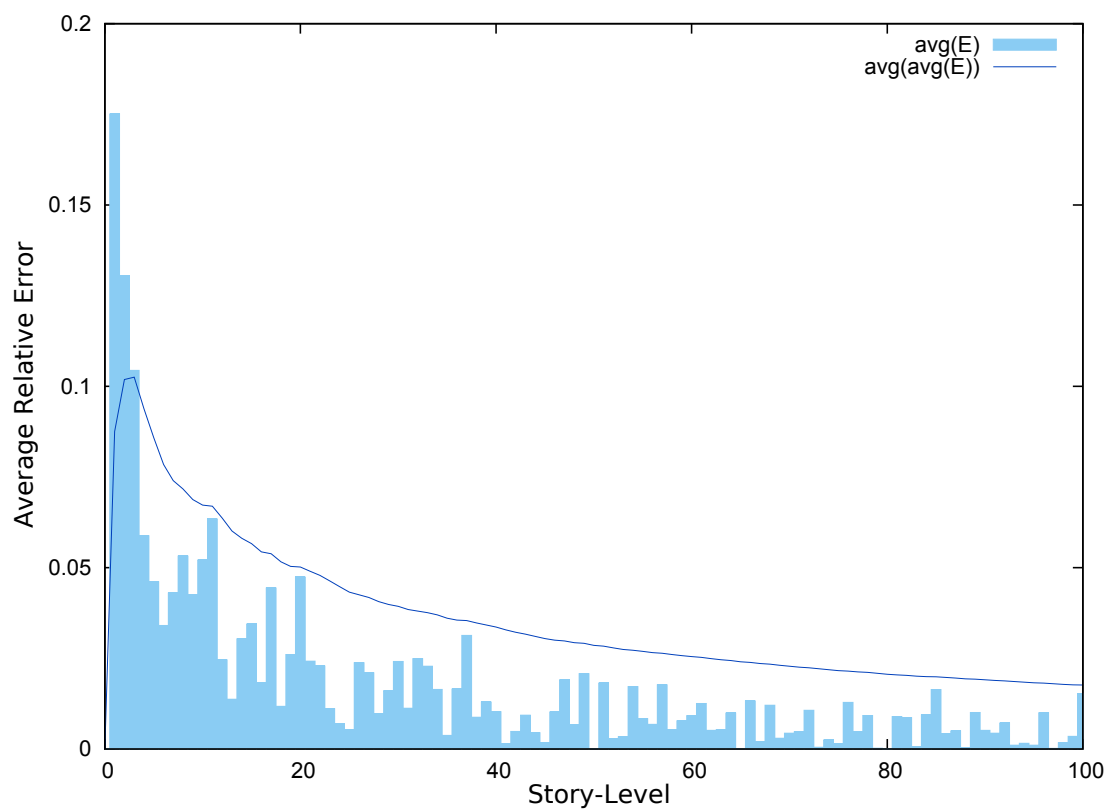


Figure 26: PaSSAGE: Relative error (average of 50 generated games).

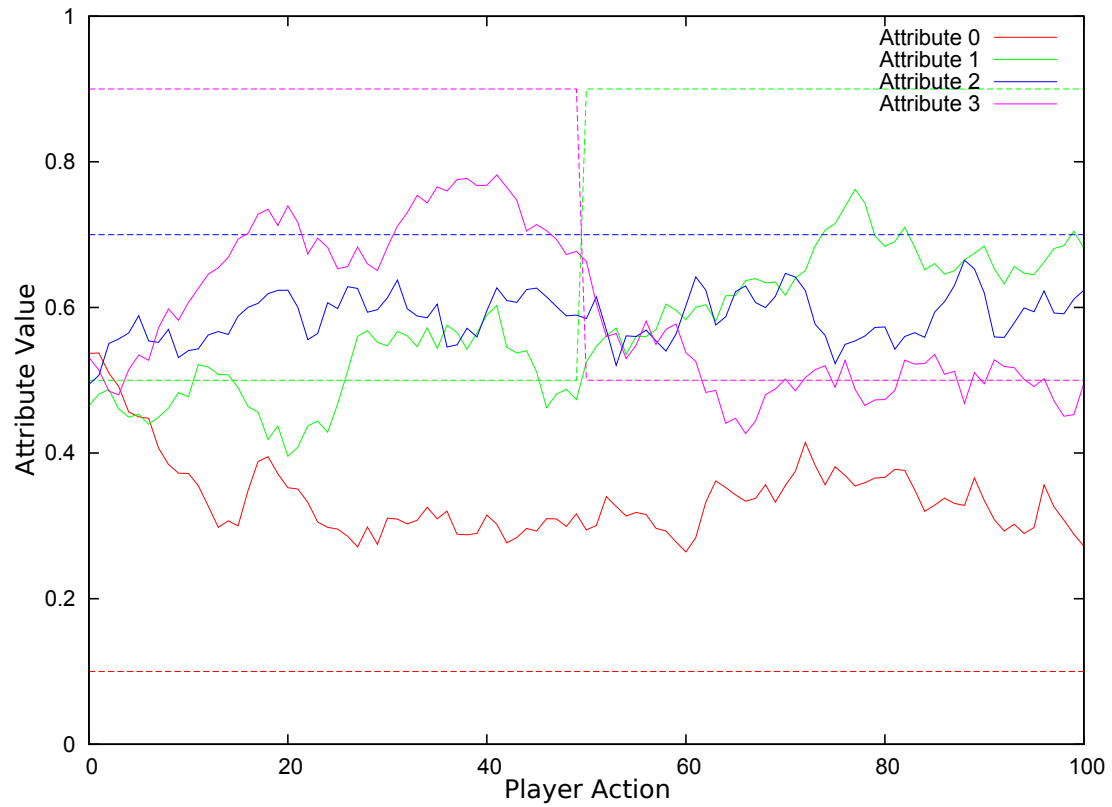


Figure 27: LMS: variable target model, Target model  $(0.1, 0.5, 0.7, 0.9) \rightarrow (0.1, 0.9, 0.7, 0.5)$ .

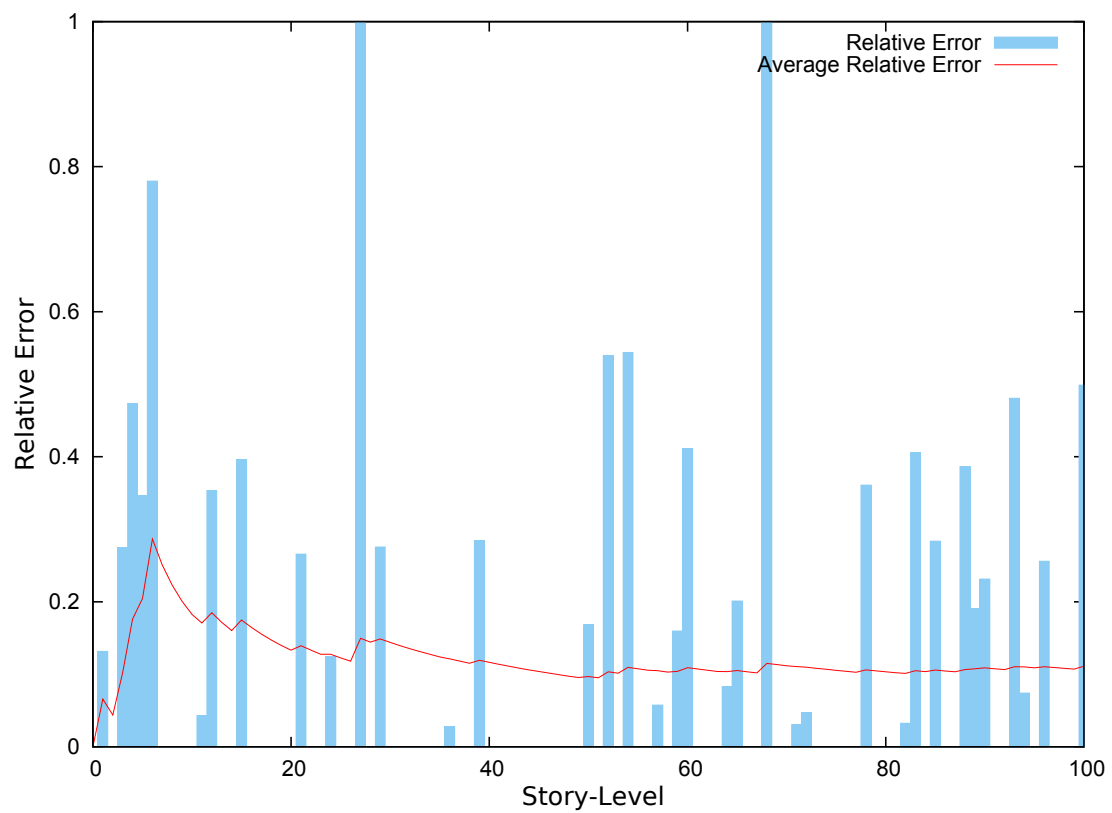


Figure 28: LMS: relative error, Target model  $(0.1, 0.5, 0.7, 0.9) \rightarrow (0.1, 0.9, 0.7, 0.5)$ .



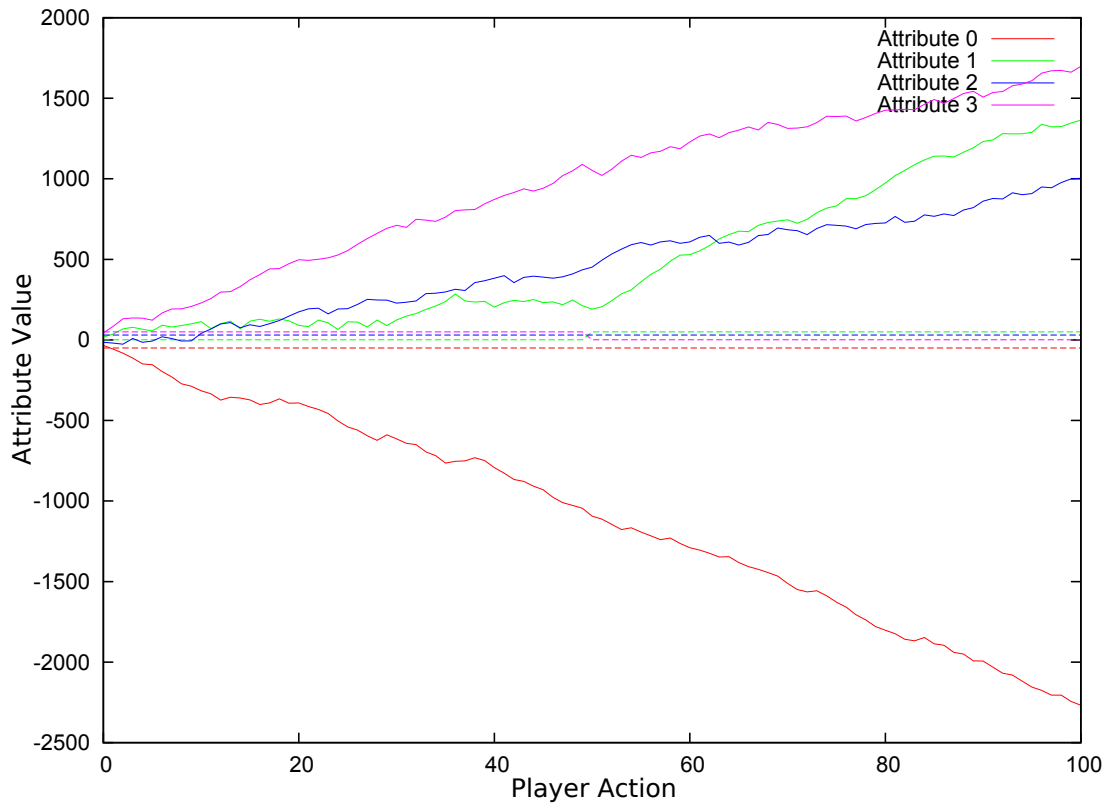


Figure 29: PaSSAGE: variable target model, target model  $(-50, 1, 30, 50) \rightarrow (-50, 50, 30, 1)$ .

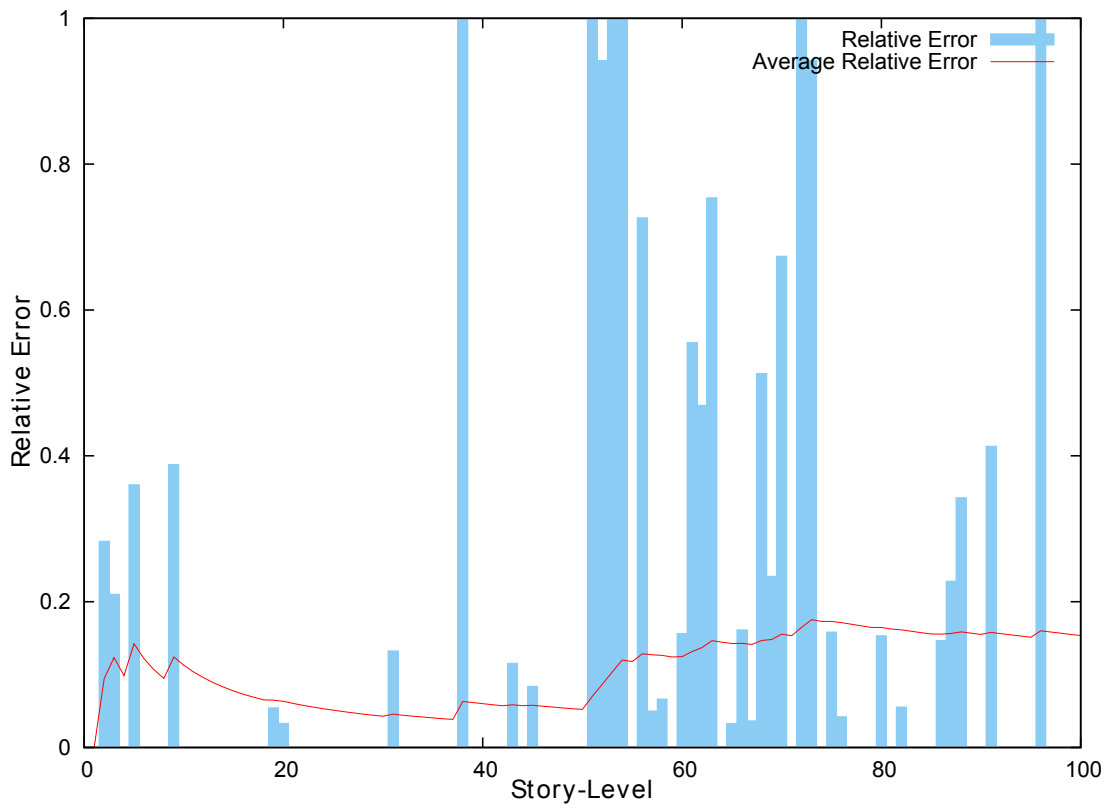


Figure 30: PaSSAGE: relative error, target model  $(-50, 1, 30, 50) \rightarrow (-50, 50, 30, 1)$ .

game play, the PaSSAGE approach should be preferred to LMS. However, this result would have to be validated empirically since during a test with real players, the governing factor could be either the flexibility of LMS or the convergence of PaSSAGE.

### 5.4.3 Adaptive Choices

The main means by which the player model influences the game lies in free transitions. Whenever a free transition is triggered, the player model is evaluated by comparing the values for each dimension along which adaptation is carried out and the most fitting scene is chosen. Since the three dimensions could be in conflict, we propose to derive a value from the weighted average of the three individual values. The three weights for storytelling, learning and playing indicate the importance each aspect is given in the current context. For example, the importance of learning could be valued the highest for a game played during a school lesson, while a game played by pupils in their free time should be more motivating and therefore focus on storytelling and playing.

We define  $w_N$  as the weight for the narrative value,  $w_G$  as the weight of the player model value and  $w_L$  as the weight of the learning value, with the conditions  $w_N, w_G, w_L \geq 0$  and  $w_N + w_G + w_L = 1$ .

If we have to choose from a set of possible scenes  $\{S_1, \dots, S_n\}$ , we first calculate the suitability along each axis of the NGLOB as  $f_N(S_i)$ ,  $f_G(S_i)$  and  $f_L(S_i)$ .

$f_N$ , as the narrative evaluation function, operates on the vector of narrative model attributes. If we define  $N_{\text{player}}$  as the narrative model of the player and  $N_{S_i}$  as the appropriateness of a scene, we make use of the vector  $1 - N_{\text{player}}$ . This vector describes the needs of the player, i.e. those story model steps the player has not yet seen. We derive the overall suitability of  $S_i$  as the distance between the player's needs and the scene's narrative annotation.

$$f_N(S_i) = |(1 - N_{\text{player}}) - N_{S_i}| \quad (5)$$

The value  $f_G$  results from the evaluation of the player model and the scene's annotation. In the case of LMS, we already have values in the range  $[0, 1]$ . For PaSSAGE, the values are potentially unbounded, in which case we require a normalization of the values. This is realized by mapping the lowest value to 0 and the highest value to 1, with the remaining values linearly interpolated between them. The suitability of  $S_i$  results as:

$$f_G(S_i) = |G_{\text{player}} - G_{S_i}| \quad (6)$$

Finally, the value  $f_L$  for learning is computed. Here, we differentiate between the list of associated skills of  $S_i$  and the list of prerequisite skills of  $S_i$ . The optimal scene would be that for which the player has all prerequisite skills and not all of the associated skills. Therefore, we compute the suitability  $f_L$  as

$$f_L(S_i) = \frac{f_{L,\text{prereq}} + f_{L,\text{assoc}}}{2} \quad (7)$$

where  $f_{L,prereq}$  determines the match of the prerequisite skills of the scene with the current skills of the player and  $f_{L,assoc}$  determines the match between the skills the learner needs and the associated skills of the scene.

After the computation of  $f_N$ ,  $f_G$ ,  $f_L$  for each  $S_i$ , the final value  $f(S_i)$  is computed as

$$f(S_i) = w_N \cdot f_N(S_i) + w_G \cdot f_G(S_i) + w_L \cdot f_L(S_i) \quad (8)$$

The scene to choose next is then chosen as  $S_{optimal}$  with

$$f(S_{optimal}) = \max_{i=1}^n f(S_i) \quad (9)$$

For scene pools, we have to take measures that prevent cycles from forming. This is realized by tracking the previously visited scenes and ranking those scenes previously seen lower.

#### 5.4.4 Integration into the Authoring Process

In order to make the adaptivity mechanisms shown here available to authors during the authoring process, we utilize the game model again. Scenes are annotated with metadata giving the adaptive algorithms information about their suitability in varying contexts. Actions are used during the game to update the models of the player and to trigger adaptive changes to the game.

In a first step, the metadata for scenes is augmented by annotations for each of the adaptation axes. For storytelling-based adaptivity, the appropriateness of each scene in the Hero's Journey is indicated by a percentage value for each stage in the story model. This is only the default model, our concept allows other story models such as the one by Propp to be integrated. For learner modeling, the scene is augmented with two lists with prerequisite skills (those skills required to understand the task or information presented in the scene) and the associated skills (those that are taught or presented in the scene). Since the set of skills can be changed by an author, it is not fixed for every game created.

For player modeling, each scene can be annotated with a percentage value for each of the four player types identified by Bartle. Again, this is only used as a standard model and could be exchanged for other models. Furthermore, since the values here do not have to be interpreted as player types but player characteristics in general, it is also possible to use arbitrary, game-related values here. We define dedicated actions for updating the narrative, learning and player models. These actions can then be used the same way as other actions are used in reaction to stimuli, thereby annotating the stimuli with an update of the respective model. For example, if a stimulus indicates that a player has chosen a more action-oriented route through the game when given a choice, the author could add an "Update Player Model" action to this stimulus in which the "Killer" attribute according to Bartle is increased.

Similarly, in order to trigger an adaptive scene change, an author can specify several transitions leading to different scenes and trigger the adaptive selection by executing a "Trigger Free Transition" action. This triggers the scene choice mechanism

described in the previous section with the scenes the author has specified as the input parameter. The optimal scene, according to the current context and the assigned importance of the adaptation dimensions is chosen and the current scene set accordingly.

Integration of the NGLOB concept into an authoring tool can also include other levels than the model. Concerning testing, we find that it is hard to test the effects of adaptive algorithms in an immersive game. This is due to the fact that games typically require players to overcome challenges and take part in other activities such as dialogues with characters that all require time. Even if these activities can be skipped, testing specific sections of a game is a hard task.

Secondly, due to adaptive interventions intended to be transparent to players, they are also a challenge for testing as it is hard to measure their effects. Therefore, a rapid prototyping tool as described in the following chapter is optimal for testing the adaptive structure of a game, since the reduced prototypical gameplay of the testing tool allows testing much more quickly. Additionally, the testing tool can give feedback on the adaptive algorithms in order to show the effects of adaptive algorithms. Since the basic technology, especially the Story Engine, is the same in a complete, immersive game and the prototype environment, insights gained in one environment can be transferred to the other.

## 5.5 CONCLUSION

In this chapter, an extension of the game model described in the previous chapter towards adaptive educational games has been detailed. This extension is based on the novel concept of Narrative Game-Based Learning Objects (NGLOBs) for the annotation and control of a game based on the dimensions storytelling, gaming and learning. By adding metadata for these dimensions to a scene, an adaptive algorithm can choose an optimal path through an adaptive game at runtime.

Concerning learning, we have examined models for storytelling that can be used in an adaptive game to ensure a high-quality, suspenseful story. By choosing the Hero's Journey as the storytelling model, authors can annotate scenes by the steps in the story model they are appropriate for. This information is then used to ensure that players traverse the game's story in the correct order at runtime.

For gaming, we have examined both personality-oriented and player-type based player models. After an examination of the alternatives, the choice of the player-type based model by Bartle has been explained. Using this model, scenes can be marked as appropriate for one or more type of player, allowing the adaptive algorithm to ensure that players experience gameplay they enjoy.

The learning aspect of NGLOBs is modeled using the framework of Competency-based Knowledge Space Theory, defining a graph structure of skills with relationships defining sensible ways of learning the content.

For the management and update of the narrative, learner and player models at runtime, the alternatives of using the Least-Mean Squares algorithm and the PaS-SAGE algorithm have been examined. After carrying out a study comparing the performance of the two algorithms in a simulation of a game, we have proposed the use of the PaS-SAGE algorithm due to its better convergence towards a fixed player

model. However, since the LMS algorithm performs better under concept drift, both algorithms are allowed to be used in this concept.

For finding the locally optimal choice of next scene in an adaptive game, we have presented a computation scheme with weights assigned to narrative, learning and playing indicating their relative importance in the context the game is played. By combining the valuations of a scene along each adaptive dimension, we arrive at a single value for each scene, allowing comparison and a locally optimal choice.

USING the model of educational games presented in chapter 4 as the basis for higher-level concepts, we now describe the major ways in which authors can be supported in their manipulation of the model in an authoring tool. The four author support mechanisms that are addressed are template-based authoring, collaborative authoring, model checking and iterative development.

## 6.1 TEMPLATE-BASED AUTHORING

In the following, the use of templates is explained. The term “template” refers to a concept found in many software products including those target users (as identified in chapter 3) are familiar with, such as office software suites or e-Learning tools. Examples are office software, where templates for slides or letters provide a layout and structure, with fields for users to enter relevant content.

Templates, in general, have several advantages for both novice and advanced authors. For novice users, they provide a structure that they can follow and be guided towards a useable result faster than without a template. It can encourage a productive way of using software by providing a best practice example. This positive effect can further be amplified by means such as wizards or embedded documentation.

For advanced authors, templates can be a good way of automating repetitive tasks. For example, if a game is composed of a number of tasks that always have a very similar structure, a template can automate the creation of this structure, leaving only the task of adding content to the structure to the author.

Apart from assisting in the use of the authoring tool itself, templates can also help authors that are working on an area of the game that is linked to a domain they are not trained in. For example, providing a template for a game that is based on a well-suited model for narrative (see section 2.4 for an overview of storytelling-based models) can assist an author in creating an interesting narrative even though the author has no experience in storytelling.

In the following, two forms of templates that are based on the game structure model are described. Structural templates are concerned with the arrangement of scenes, transitions, objects and action sets, whereas interaction templates encapsulate interaction between the game and the player and allow the separation between authoring and game programming in this concept.

### 6.1.1 *Structural Templates*

A structural template is a pre-configured configuration of scenes, transitions, objects, action sets and parameters. It is intended to be added to the game’s structure at a position specified by the author during the authoring process or right at the beginning of the authoring process as a template for an overall game.

The amount of structure and pre-configured elements in a structural template can differ according to its intended use. In a 2D game such as an adventure game, a meaningful small structural template could be one including one scene, a background image for this scene along with an exit that can be used to switch to a different scene. The particular values for properties, such as the actual background image or the location of the exit, are then configured by the author.

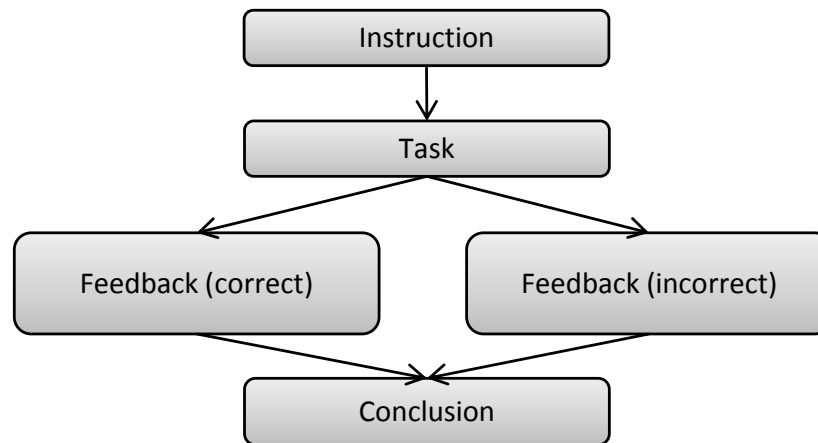


Figure 31: An example structural template, composed of scenes and transitions to form a very common pattern in e-Learning.

A larger example is found in games that feature parts resembling those found in e-Learning products. For example, this could include a task introduction, the task itself, several variants of feedback according to the way the player solved the task, finished by a conclusion. This would be modeled as a structural template consisting of a network of scenes with a central start point, the task introduction, a scene for the task, a set of branching scenes for the feedbacks that are joined again in a scene for the conclusion (cf. figure 31). This template would then be instantiated as many times as the author has use for it.

The largest possible structural template is one that is applied to the game right at the beginning of the authoring process. This kind of template can guide the work of authors and be used as a starting point that is then adjusted to the specifics of the authored game. An example is to model the stages of the Hero's Journey (see section 5.1) as scenes which are pre-configured to have the appropriate metadata for adaptation based on narrative (see previous chapter) in place. This can guide an author who is not familiar with the structure of suspenseful narrative in structuring the story of the game in a meaningful way.

### 6.1.2 Interaction Templates

The intention of this concept and the utilization of the game model lie in empowering authors to create games despite complex underlying technologies. Therefore, the authors have to be able to work with the authoring tool in an abstract way that, nevertheless, results in a complete game running on a game engine. The basis for this work is the Story Engine described in the previous chapter, as it manages the execution of a game at runtime. Despite the support for visual programming that is afforded by the action set concept, authors would not be able to program complex

game mechanics or foundational techniques (such as 3D character movement) using this technology as the languages would either be too restrictive or the result would be complex and unintuitive.

Since we target non-trivial games, these game elements and foundations have to be provided by programmers. In the following, we propose a concept for components in game authoring tools and game runtime environments that provide certain gameplay elements to authors. Since these templates are not only concerned with structures as the previous category of structural templates, we refer to them as interaction templates. Thereby, the game model in conjunction with the available templates defines the design space of games possible to be authored by an author. This space is defined by the affordances of templates, the possible customizations that they allow and the ways in which they can be combined by making use of the game model.

To provide an example, we use a case encountered in the creation of a virtual city rally for the city of Darmstadt. The game's narrative is inspired by the thriller story and has the player moving around the town, collecting knowledge and answering questions about the city. The finale of the game is a "word puzzle" in which the player has to arrange letters in the right order to form the code word (see figure 32).

If such a game were to be created in a standard game engine by a programmer, it would entail writing the code for the graphics of the game (rendering letters to the screen at their current location) as well as that for interaction with it (interpreting input signals from the mouse device as switch signals, checking for the correct end state). This would be out of reach for an author as characterized in chapter 3. Using the approach of the authoring tool outlined in chapter 4, this mini game could be implemented using images and an ActionSet for controlling the game. However, this would require a large number of individual objects (one for each letter) and many variables and conditions, resulting in complicated ActionSets.

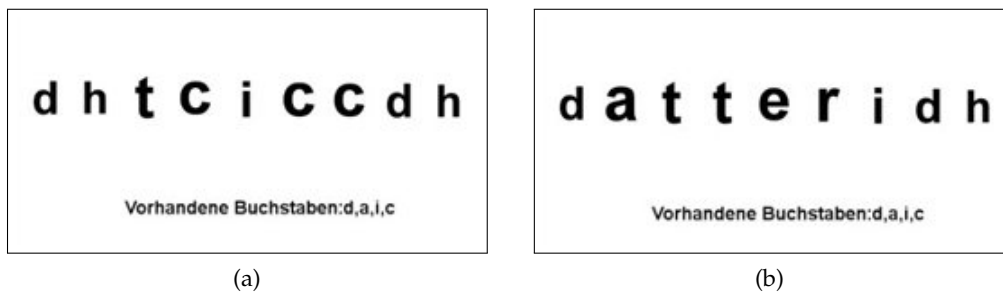


Figure 32: A simple example of an interaction template. The encapsulated gameplay is that of a "word puzzle": by clicking on each letter, the letter is cycled through a set of letters. When all letters are correctly set, the correct word is displayed and the puzzle solved.

On the other hand, assuming an interaction template for this game mechanic was created, an author would only need to add the template to the corresponding scene. In a **WYSIWYG** fashion, the author could enter a target keyword which is then used to create the individual letters. By setting up parameters such as the difficulty levels, the author could directly influence the permutations of letters. Instead of many individual events to be handled by the author, one logical high-level event for this game is apparent, indicating that the player has successfully finished the objective by putting all letters at their correct position. During gameplay, all low-level inter-



action such as switching a letter when it is clicked could be handled by an author. Furthermore, by exchanging an interaction template by another one complying with the same set of specifications but realizing it differently (e.g. in 3D instead of 2D), the game could be changed without changing the game content.

In order to allow an author to create a game consisting of tasks such as the one described above without having to program, we propose the concept of an interaction template. A fitting counterpart from the world of analogue games are “frame games”, as described by Stolovitch and Thiagarajan [139]. By this term, they refer to games (for example board games or card games) that feature a certain set of rules but whose “content” can be adapted. For example, the cards of a card game could be marked with vocabulary to be learned, therefore involving the task of learning them into the game. Therefore, a game is used as a “frame” for learning.

From a technical perspective, we can turn towards Component-Based Software Engineering (CBSE) [141], referring to a common software development practice in which systems are composed of maximally independent component which carry out clearly defined, atomic tasks. The positive features of components include the following:

- Well-defined interfaces: Interfaces of components are strongly separated from the implementation of the component and give a complete view of only those features that have to be exposed to other components in the system.
- Self-describing: Components can be described by their interfaces which include all the offered functionality of the component
- Separation of concerns: Communication between components is restricted only to the data necessary for the overall system. Apart from this, all components can complete all their tasks on their own.
- Substitutability: Due to the strong separation of interface and implementation of a component, components can be exchanged, for example to satisfy non-functional requirements such as speed or memory restrictions. This can also be used to produce different versions of the system for various platforms.
- Tool support: Tools for the composition of systems by non-programmers using the CBSE approach have been developed, for example by manipulating and connecting visual representations of components. One example is the connection of the output of one component to the input of another component.

In game development (especially concerning game engines), CBSE has been used to some extent. While most applications of the approach are only found in commercial products, some research on CBSE game-development has been published. Folmer [43] present a general reference architecture for game engines which is based on components.

In the domain of e-Learning, El Saddik [40] provides a model of configurable educational components.

We focus here on the usage of CBSE in authoring tools, especially the possibilities for author support that the realization of interaction templates in a component-based fashion allows. The work by Maciuszek et al. [90] on an intelligent tutoring system for educational games, based on component-orientation, is an example in this field.

Similarly Bisognin et al. [13] show an educational system which uses components to encapsulate parts of a game. The EDoS authoring system [151] was created to target this game architecture and therefore uses authoring concepts aligned with component-based games.

The e-Training DS authoring system [147] (designed to run on the Nintendo DS platform with very limited computing power) incorporates the ideas of CBSE in a comparable work to that presented here. However, the authoring tool only allows the configuration of educational components and does not provide the possibility of connecting components with the rest of the game as is allowed in our concept.

In the following, we describe our approach to integrate interaction templates into the authoring process using the basic model described in chapter 4. Figure 33 is an overview of the aspects an interaction templates has.

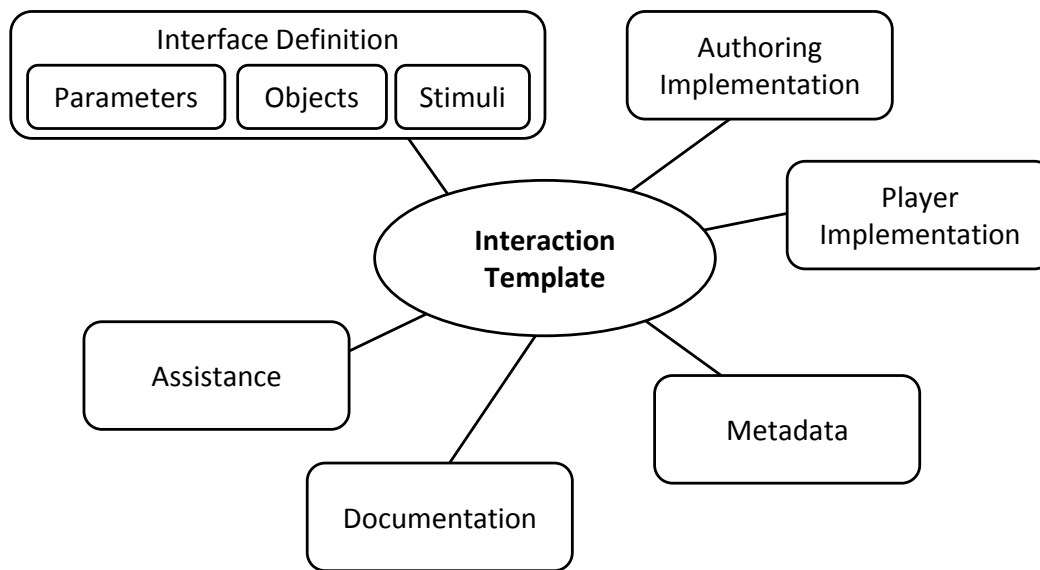


Figure 33: The conceptual overview of an interaction template and the associated aspects.

### *Interface Definition*

Consistent with component-based software engineering, interaction templates have a well-defined interface definition that describes the ways they can be configured. This interface definition here is to be understood as a definition concerning the game model described in chapter 4, which is separate from the underlying software interface definition in an implementation.

The interface definition for an interaction template contains definitions of the involved parameters, objects and events. The uses of the interface definition are shown in figure 34.

Parameters are used to configure the details of the interaction template. While some characteristics of the interaction template will not be configurable by an author in order to limit their complexity, others, especially those related to the content and game mechanics of the interaction template are exposed to be changeable by an author. For example, a parameter for a “puzzle” interaction template is the image to cut into puzzle pieces and allow the author to interact with. In this case, other

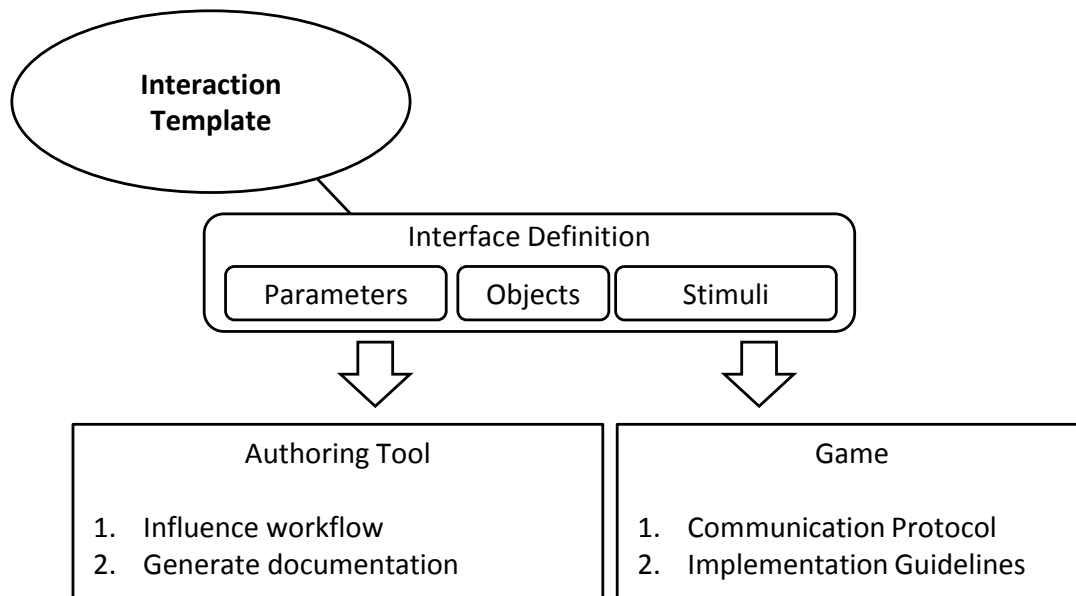


Figure 34: The main uses of an interaction template's interface definition

parameters, such as implementation details such as the initial random configuration of the puzzle or the pattern of puzzle pieces could remain hidden from the author.

Objects defined by an interaction template are those objects which the interaction template interacts with during the game. As an example, consider a “mark difference” interaction template in which a player has to identify and click on differences in two almost-identical images. The interaction template could define a “difference marker”-object which the author can instantiate a number of times (one for each difference in the image). During play, the interaction template monitors the state of all markers and only triggers the “game solved” stimulus when they have all been clicked.

Events of an interaction template are the stimuli this interaction template can trigger. Staying true to the high-level nature of stimuli, they are to be used only for those events that are necessary for the author. Prime examples are stimuli that indicate that the player has finished the interaction template's “goal” and the game should move on. Based on the stimuli, an author uses the ActionSet mechanism of the authoring tool to define the actions (e.g. switching to a new scene) that continue the game.

The use of this interface definition has several benefits. During implementation, programmers who implement the component can use the definition as a set of requirements that have to be fulfilled to correctly implement the template. On the other hand, during authoring the authoring tool can use the interface definition to automatically generate documentation or wizards to guide novice users in using a template.

### *Authoring Implementation*

Based on the definition of an interaction template, two implementations are necessary. The first, which we describe here, is the authoring tool implementation. This implementation is used in the authoring tool to allow an author to configure the interaction template. It provides a graphical user interface to the objects it defines

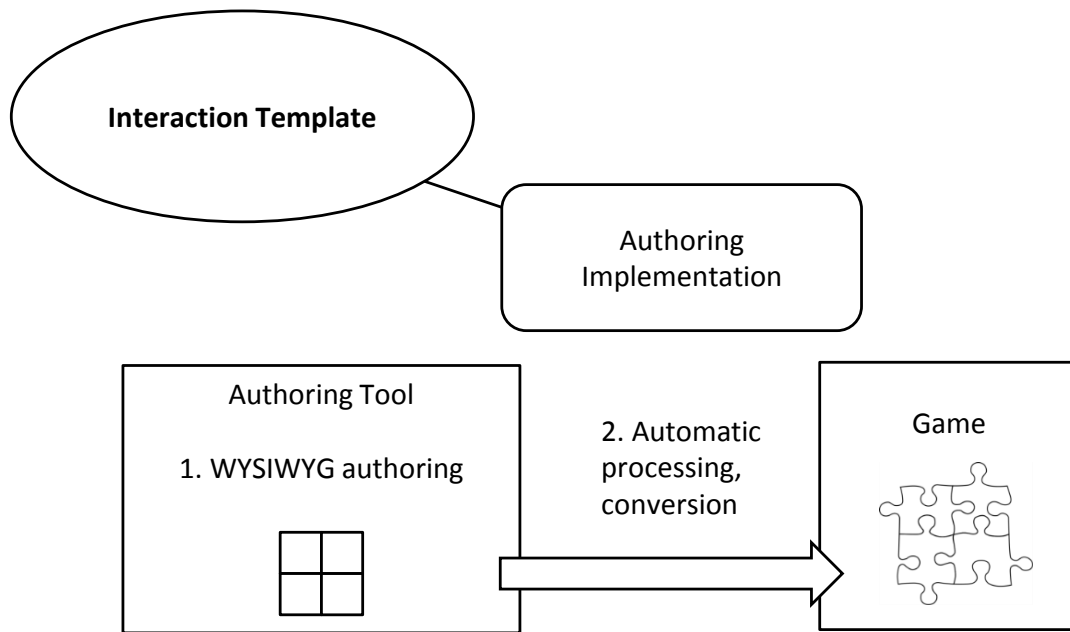


Figure 35: The main uses of an interaction template's authoring implementation.

and gives a visual preview of them. Figure 35 displays the context of the authoring tool implementation.

The specifics of this implementation are linked to the nature of the interaction template. For example, some interaction templates define new types of objects that can be placed in a WYSIWYG-environment in the authoring tool, while other interaction templates give only a preview of their content. A “puzzle” interaction template, which is configured using one input image that is subsequently cut into a puzzle pattern could show the image, overlaid with the pattern of the puzzle to give the author a sense of the look of the puzzle in the game.

Apart from the interaction with the author, the authoring implementation of an interaction template can automate tasks for the author. Using the “puzzle” example again, the task of creating individual puzzle pieces as images from the larger input image could be carried out in the authoring implementation of the interaction template, in order to save resources on a target device.

Without a specialized user interface provided by an authoring tool implementation, authors could only edit parameters of an interaction template, without the possibility to see a preview and get immediate feedback for changes made.

The cohesion between the authoring tool framework and the interaction template should be reduced to a minimum. Therefore, interaction templates are ideally implemented as components which are loaded into the authoring tool during runtime, so that the authoring tool does not have to be adapted whenever a new template is added. Furthermore, this helps in making the implementation task of interaction templates easier since templates are not hard-coded into the authoring tool and implementers do not need to know the internal details of the authoring tool.

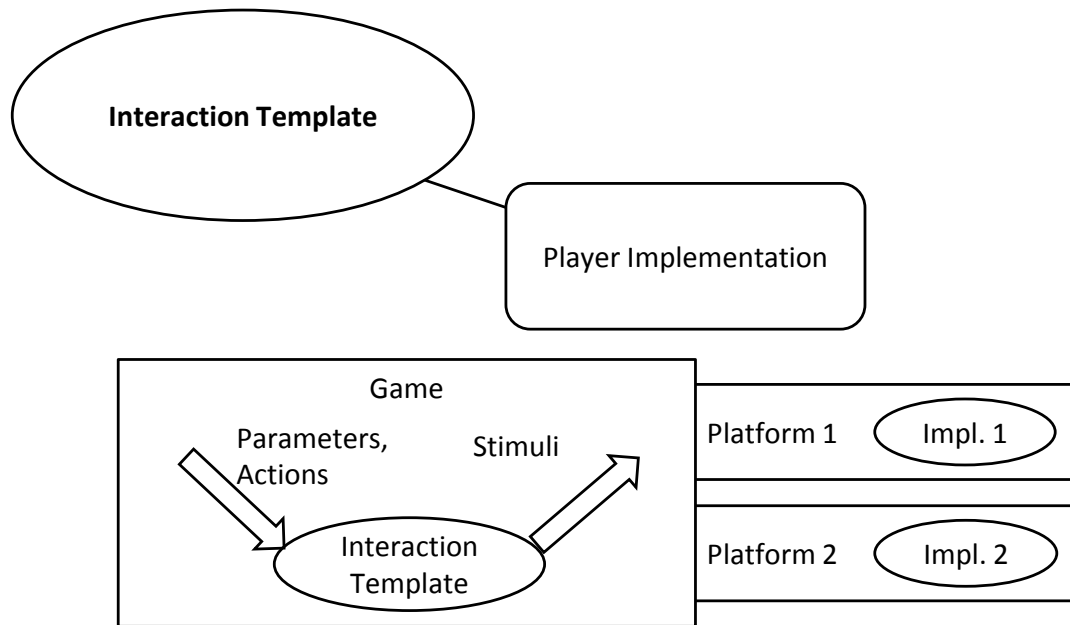


Figure 36: The main uses of an interaction template's player implementation

### *Player Implementation*

The second implementation of the interaction template is the implementation in a player application. This implementation is the one a player interacts with during play. Therefore, it is where the actual game mechanics of the interaction template is implemented. It uses the configuration data (parameter values and objects) from the interface provided as parameters to set up the gameplay specifics, it displays and manages objects, tracks the current state of the game and triggers events (e.g. user inputs) whenever they occur. This implementation is realized by game programmers, since it involves handling user input, rendering subsystems and other low-level functionality that requires game programming skills and knowledge of the target platform and operating system environment. Figure 36 illustrates the use of the programming implementation.

Since interaction templates are intended to be limited to only small parts of games (scenes), and since their definition (concerning the interface and the intended game mechanics) are clearly defined, programmers can work on them in relative isolation when implementing them and have clear guidelines what the implementation should do.

Concerning portability and cross-platform publishing to different devices, the player implementation of interaction templates is the central point for programmers' work. In order to port a game to a new target platform, the interaction templates it uses are required to be ported, making the task of porting the game again well structured. Necessary adaptations to the target platform, for example concerning differing input mechanisms or other platform-specific changes, can be made in these implementations. Situations in which this is necessary are porting to mobile devices with smaller screens or to devices with touch screens vs. mouse and keyboard input. However, since all implementations are driven by the data file created in the authoring tool, portability of the overall game is facilitated by porting the re-usable templates to a new target platform.

### Metadata

Apart from the interface definition, further metadata can be added to interaction templates. For example, the NGLOB concept of appropriateness of scenes to a certain context of narrative, learning and gaming models can be utilized by also defining the appropriateness of interaction templates. For example, interaction templates could be rated concerning their appropriateness for certain player types, such as templates allowing interaction with characters which are well suited for “socializers”. Concerning learning, templates could inherently be connected to some skills. For example, an interaction template implementing a game which follows the rules of physics would be well suited for learning about effects such as gravity or conductivity.

During authoring, this information can help guide authors towards using the appropriate game mechanics, and an automated analysis (see section “Model Checking” below) can make authors aware of mismatches.

### Documentation

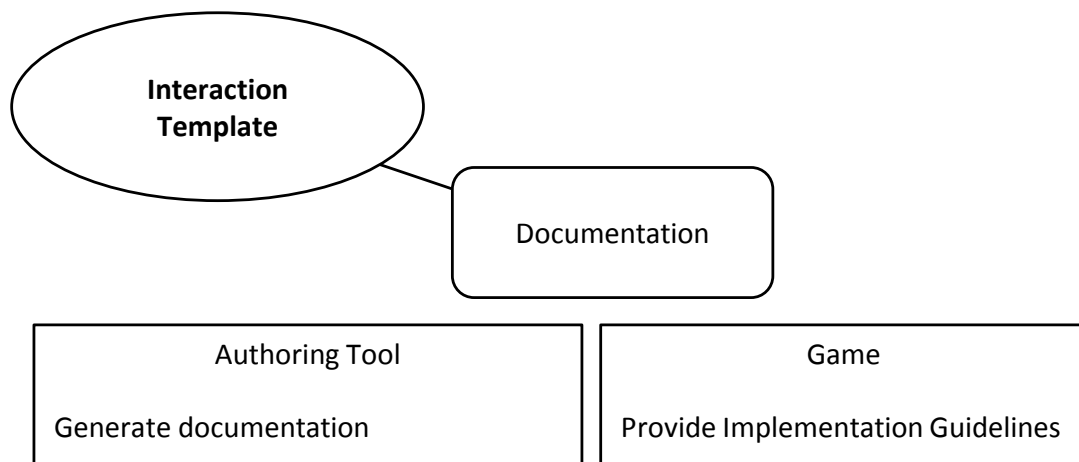


Figure 37: The main uses of an interaction template’s documentation.

Documentation for interaction templates can be separated into two categories: manually created documentation and automatically created documentation.

Manually created documentation includes textual descriptions of the interaction template itself such as intended gameplay or implementation details, as well as the involved parameters, objects and events. Furthermore, it can include screenshots or videos taken from already existing implementations, providing authors with a preview of the final product in the game. Documentation is used by programmers implementing the gameplay template on the one hand and authors using the implementation on the other hand. A schematic overview of the documentation’s use is shown in figure 37.

Automatically created documentation is based on the interface definition of the interaction template and the metadata. It can give an author a quick overview of the template’s structure including optional and mandatory parameters by listing these items in a short form.

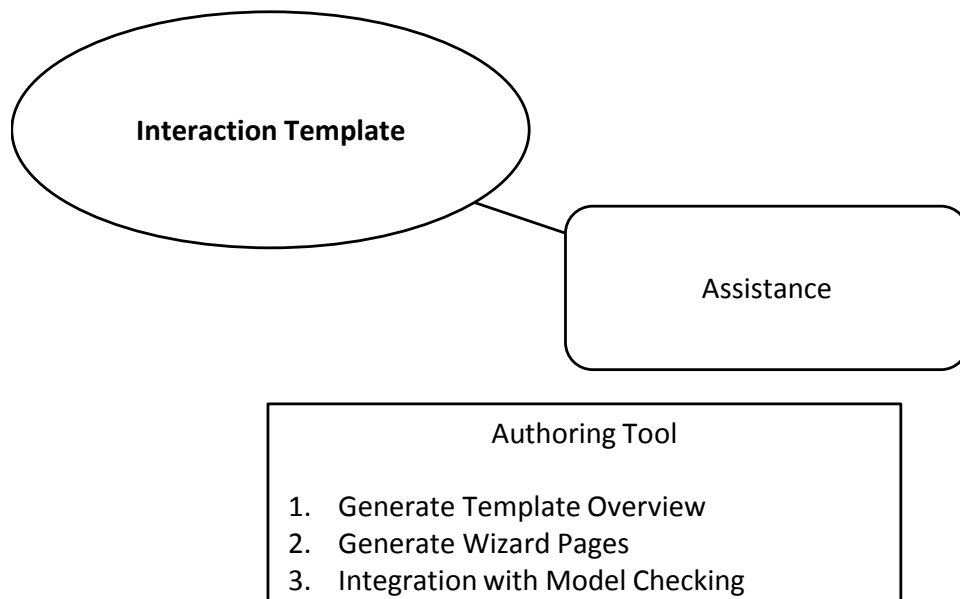


Figure 38: The main uses of an interaction template's assistance aspect.

### *Assistance*

Based on the interaction template's definition, metadata and documentation, it is possible to assist authors in the correct use of the interaction template. This starts at the choice of interaction template for a specific scene. The authoring tool can create a list of available interaction templates and present them to the author. Figure 38 visualizes the assistance-aspect of interaction templates.

After an interaction template has been chosen, two methods of assisting the author are possible, differentiated by the level of freedom they give authors and conversely the amount of guidance they provide. The more structured and strict form of assistance is in the form of software wizards that guide the author through the authoring process using the interaction template. This software wizard can be created automatically from the interface definition and documentation. Necessary parameters and objects to be created can be shown to the author along with their documentation. In this process, it is less probable that authors commit structural errors. Furthermore, it helps novice authors in understanding the correct usage of interaction templates. The settings made in the wizard are reflected in the authoring tool, so that advanced authors can then later fine-tune them. Expert users who do not require the wizards anymore because they can work more efficiently themselves can disable them.

The second, less intrusive method of guidance for more experienced authors lies in checking the structure of the interaction template during authoring, making the author aware of errors such as using the wrong object types for the interaction template or forgetting to fill out essential fields. This is integrated with the model checking capabilities shown in this chapter.

## 6.2 COLLABORATIVE AUTHORING

In order to support use cases in which several users work on a game together in the authoring tool, we examine how collaborative authoring can be supported in an authoring tool. We find that two kinds of collaboration are relevant in this context:

*Direct collaboration* is used here to refer to collaboration encountered during the development of a Serious Game, in which the members of the development team (programmers, designers, artists, domain experts, ...) are collaborating, which was shown in chapter 3 to lead to communication overhead and a resulting loss of content in the game. This type of collaboration therefore is characterized by users who are working closely together (also concerning temporal and spatial aspects).

Possible means of collaboration in game development teams are identified by Tran and Biddle [152]. Using the approach of this thesis, we can support two of the collaboration methods described Tran and Biddle. All members of a team are able to work in the authoring tool and see the whole structure of the game as it evolves. Therefore, a team-wide vision and understanding of the game can be fostered. A bottleneck identified is the need for programmers to be involved closely in all parts of game development. Our approach of encapsulating game mechanics implementations into interaction templates concentrates all involvement of programmers in this task, thereby removing this bottleneck. Furthermore, instead of a large set of different tools being used (text editors, spreadsheets, programming IDEs etc.), work is done mostly in the authoring tool. The authoring tool therefore resembles a “virtual blackboard”, allowing all team members to see their own activity and that of other authors.

*Indirect collaboration*, on the other hand, is epitomized in the third scenario identified in chapter 3 where an expert (e.g. a teacher) is provided with a complete game and configures it for an individual user or a set of users. In this case, the developers of the game have to provide means to personalize the game and the domain expert has to be empowered to configure the game using the authoring tool. Temporally and, in most cases, spatially, the first game-expert author (creating the game) and the domain-expert author (configuring the game) are separated.

For the latter form of collaboration, we propose the mechanism of a domain-specific mode in authoring tools using the model described in chapter 4. While operating in the normal authoring mode, the author using the tool is completely free in the authoring process. Structures can be created and filled with interaction templates and further logic. During this process, the author can additionally assign metadata concerning the use in the domain-specific mode to items in the game (scenes, objects, variables). These metadata are concerned with accessibility of model elements in the domain-specific mode:

- *Read-Write*: Items will be visible and changeable for a domain expert
- *Read-Only*: Items will be visible but not exchangeable by the domain expert
- *Invisible*: Items are completely inaccessible to the domain expert

Figure 39 illustrates the main differences between the two authoring modes. Whereas the specialist game author, whose view is shown in figure 39a, works with all possibilities of the authoring tool, the domain expert, whose view is shown in figure 39b,



only has access to those elements deemed important by the first author. Elements such as scenes, transitions and objects with a padlock are the read-only elements, whereas the stippled objects are those marked as read-only.

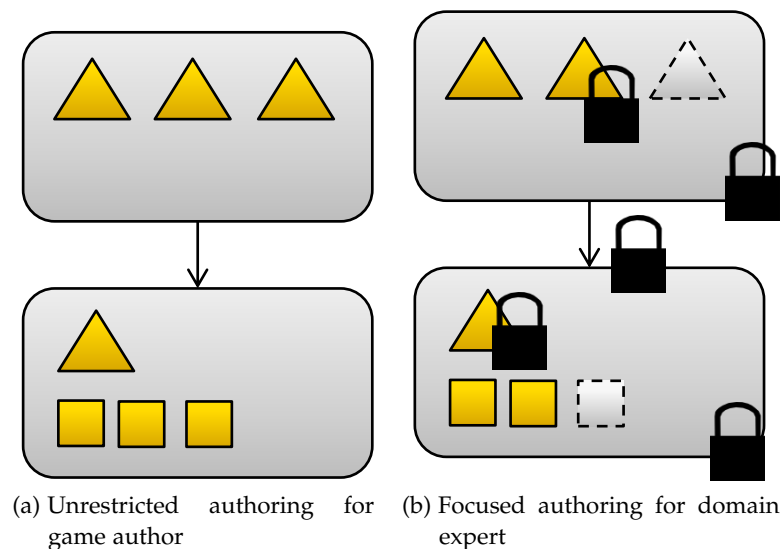


Figure 39: Comparison of views depending on authoring mode.

Furthermore, the ranges for items can be limited. For example, for a string parameter, the set of possible strings to be entered can be limited. Instead of a free input field, a domain expert would find only a drop-down text box with the possible strings.

In the domain-specific mode, authors are only presented with the relevant information filtered by using the accessibility settings previously made. Additionally, adding scenes and transitions can be restricted.

Note that this second, indirect collaboration mechanism can also allow authors to define and prototype interaction templates before they exist. By specifying the relevant parameters that will be used to configure the interaction template and building a visual representation from simple multimedia objects (e.g. images or text), these prototyped interaction templates can act as placeholders until the interaction template has been created on the one hand and as a specification for interaction template implementers on the other hand. Until an authoring implementation is created, the inputs made in the prototyped template can be used as test data for the player implementation.

See Appendix A.10 for an example of this method of interaction template creation in the context of a game for health, where the domain-specific author is a therapist configuring a game for a patient. During authoring in the domain-specific mode, the expert only has to change the parameters relevant for the health-aspects of the game, e.g. target heart rate or other vital target parameters.

### 6.3 ITERATIVE AUTHORING

By iterative authoring, we refer to authoring that is carried out in several cyclical phases involving early prototypes for testing and continual refinement. This can be

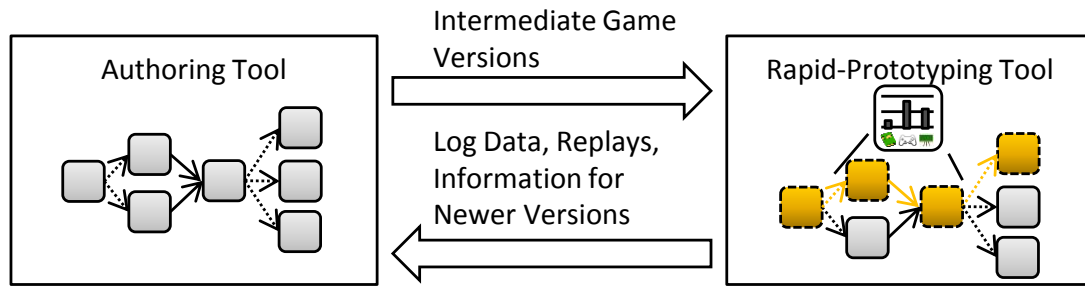


Figure 40: The cyclical workflow provided by iterative authoring. The stippled scenes indicate those scenes actually visited during the test session.

seen as the authoring equivalent of agile approaches that are used in game development (cf. chapter 2). An author creates an early version of a game by using placeholder content or descriptions of scenes instead of content. This version can then be tested by the author himself or herself (for example by verifying whether adaptive choices in the game are made as he or she intended them to) or by testers who give either automated feedback (by keeping log data) or manual feedback by reporting to the author. The knowledge gained about the game can then be used in consecutive authoring iterations.

Iterative authoring is enabled in this concept using two related mechanisms, that of using a testbed for testing out the effects of algorithms (such as the effect of changing metadata for adaptivity) and that of rapid prototyping. This is in line with the call by Musil et al. [101] for an agile approach that promises to improve the production of games.

Figure 40 illustrates the cycle of testing and authoring when an authoring tool is supported by a rapid-prototyping tool. This tool allows testing of games that have not been finished yet, yielding information about the effects of player choices, for example on the adaptive algorithms as described in chapter the previous chapter. This information can be fed back in the form of automatically gathered data such as logs and playbacks that the author can view, or manually by watching testers make use of the rapid prototyping tool and then making changes to the game.

Note that iterative authoring could also be used to create extensions to already existing games, in the case of re-authoring as described in chapter 2. Extensions to the game can be tested prototypically to assure their appropriateness and be implemented more efficiently.

Since the control over a game modeled using the approach of this thesis is carried out at a high level of abstraction, games can be created without complete implementations. For example, interaction templates can be filled with placeholder content or interaction templates themselves can be prototyped as shown above. A testbed tool would only have to simulate the stimuli created by the interaction template and log any output of the Story Engine which interprets and executes the game model.

Assuming the existence of such a rapid-prototyping and evaluation tool, we find the equivalence of a testbed in other areas of computing. Testbeds are commonly used in the functional evaluation of algorithms. For some adaptive storytelling and game systems, testbeds have been used to evaluate how a given algorithm performs under various inputs. The Remote-Controlled Environments Interface (RCEI), described by Peinado et al. [108], defines a set of commands which connect an interac-

tive storytelling system to a virtual environment, which is then used as a testbed for the storytelling system.

Bailey and Katchabaw [6] describe the evaluation of dynamic difficulty adjustment algorithms using a testbed.

In a similar fashion, rapid prototyping refers to a practice in which early prototypes of a product such as a game are created early in the production process in order to be able to test the product at this early stage and to use the gained insights early while they can be made easily and with low costs. For a description of the role of rapid prototyping in the game design and development process, see the account of Fullerton [45, ch. 7] on this topic. Coiana et al. [30] describe a system for play-centric design by means of rapid prototyping of games on mobile devices.

During a test using such a prototype game, information can be logged and then used by an author to make informed changes or additions to the game. Information to be logged can include the actions of the players, timing information such as the amount of time spent in each scene or the evolution of adaptive models (player, learner models). For example, the lengths of time that players spent in a scene could be averaged from a set of log files and then be used in the game, for example by providing hints whenever a player takes substantially more time than an average player.

Apart from prototyping the game with interaction templates already in place, the iterative design process could also be applied to interaction templates themselves. An initial version could be created using the domain-specific authoring mode described above. First implementations which realize the gameplay in a prototypical fashion (for example, only simulating the necessary stimuli that the interaction template defines in its interface) could then be used to test the interaction template in the context of the game. This method of testing is allowed by the concept defining the Story Engine component that is only required to react to stimuli from the game and not interpret any detailed data by the game. Therefore, the only necessary events that have to be generated by an interaction template prototype are these stimuli, which can be achieved easily by providing buttons that trigger each stimulus. Of course, such a sparse representation can only be used to test the effect of the interaction template on the rest of the game and not the appeal to an actual player. Finally, the actual development of the template would start when the design was verified using these early tests.

#### 6.4 MODEL CHECKING

“Model checking” is the umbrella term for techniques that can automatically verify the correctness of a model in comparison to a set of specifications [28]. Since we have presented a game content model in chapter 4, we can apply model checking to a game model created during the authoring process, in order to support authors with checks for syntactic and semantic errors. For example, such a method can warn the author that a dead end (a section of the game that can never be left because of conflicting or missing game logic) exists.

Figure 41 shows the integration of a model checker into an authoring tool. The model described in chapter 4 underlies the user interface of the authoring tool and all author decisions are mapped to this structure. The authoring tool automatically

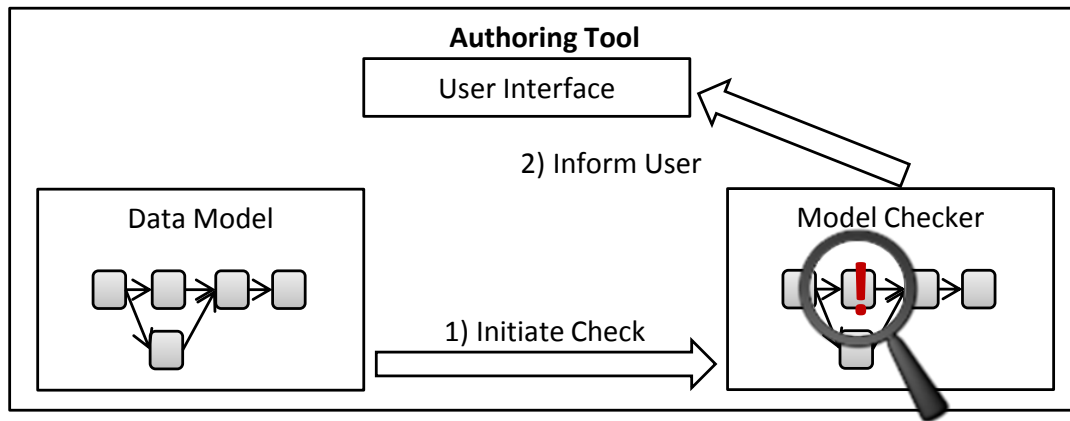


Figure 41: The model checker as part of the authoring tool.

triggers the model checker to analyze the model, possibly resulting in warnings for the author with a reference to the location of the cause of the warning.

Few related work from the field of games exists. One example is the account of Moreno-Ger et al. [100], who describe an extension to the e-Adventure authoring tool which uses model checking to inform authors of potential problems in the game.

The structured input and configuration provided by using the model described in chapter 4 diminishes the possibilities for making syntactical errors by authors. Comparing the model to a programming language, authors can make considerably less syntactical errors. For example, assuming that the authoring tool's implementation of the model is without bugs, all structures that authors create will be compatible with the Story Engine as the interpreting unit by definition. In a programming language, on the other hand, users could miss-type keywords or omit brackets or semicolons necessary for a parser. The remaining errors that authors using the model can make are on a semantic level, and are suitable for model checking. The first possibility for finding such errors early in the authoring process lies in using the iterative design approach shown above. Model checking, on the other hand, can make authors aware of errors immediately without the need for testing the game.

Several categories of semantic errors can be checked, some of them are listed below:

- Errors related to transitions: For example, an author might have created a transition in the game between two scenes, but forgot to add an action to trigger the transition in the first scene. The next level of error is that the transition is triggered when a condition is true, but the condition itself can never be true in the game, for example due to a variable referenced in the condition never being set to a required value.
- Unused elements: While not necessarily an error, authors can add objects (especially logical elements such as variables) and forget to use them in the game.
- Error related to interaction templates: For example, in an adventure game, a virtual character would have to be marked as the player-controllable main character. If this was forgotten, the game might not work as intended by the author.
- Errors related to adaptivity: As described in the previous chapter, authors can annotate scenes with narrative, learning and playing information in order to

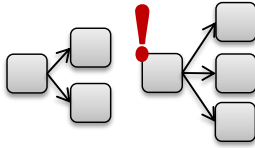

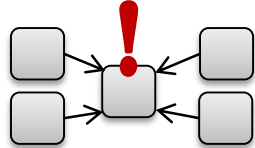
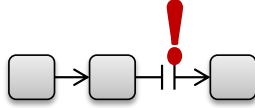
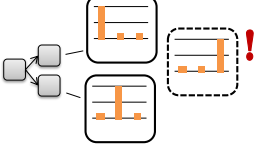
| NAME                    | DIAGRAM  | APPLICABLE METHOD                       |
|-------------------------|--|---|
| Islands                 |   | Graph analysis → Search algorithm       |
| Unused Variable         |   | Search for the variable                 |
| Dead End                |   | Graph analysis → Search algorithm       |
| Unreachable Scene       |   | Data flow analysis, test satisfiability |
| Player type unaccounted |  | Data flow analysis for player type      |

Table 7: An overview of possible model checks and applicable methods.

make the game adaptive during runtime. These data can be used to check for semantic errors or problems. The narrative information in the annotations can be used to determine if authors have created scenes which span the whole of a chosen story model. The play preference information can be used to infer whether a certain player type has been preferred by the author. The information about learning content can be used to check if all learning elements in the game's curriculum have been covered. Furthermore, the model checker can analyze whether appropriate learning paths through the game exist or if, for some players, no adequate paths exist.

In all these cases, a model-checking approach could issue warnings and provide suggestions for fixing the error. By using contextual information, these warnings can be issued in natural language, describing the malformed situation and providing a suggestion for handling it.

Table 7 provides further examples of conditions that can be checked, along with information about the applicable methods to find the respective problem. As we can see, several basic algorithms are applicable in many of the cases. Graph analysis, for example by search algorithms, can give information about the organization of the game model and find problems such as islands which are constituted by scenes that are not connected to any other scenes. Data flow analysis, similar to the analysis of

computer programs, is necessary to give information about possible values of variables. This information can be used when checking whether a condition is satisfiable at a certain point in the game. In this context, the satisfiability of a condition has to be checked.

A first small-scale experiment hints at the usefulness of the model checking approach. In this experiment, a version of the authoring tool StoryTec including model checking capabilities was presented to the participants who had to identify five errors that had been introduced in a game. The study included 6 male participants aged 23 to 33 who claimed to have worked with StoryTec before. The study was carried out under supervision in the context of the thesis prepared by Deborah Buth (see appendix D for supervised theses). The results have been reviewed and are reproduced here.

The prepared game included five errors from the following categories:

- Unreachable scene
- Malformed action
- Malformed condition
- Unused transition
- Unused variable

Each participant was given the task of finding all errors in the game and correcting them. The test supervisor monitored the test and noted the times spent on finding errors and the order in which errors were found. Initially, the participants were given the prepared game and a version of StoryTec including no model checking capabilities. After a set time of 12 minutes or after the participants claimed to be unable to find all errors in the game, the supervisor introduced a version that included model checking and instructed the participants to attempt to find the remaining errors. Afterwards, the concept of model checking was explained to the participants and they were handed a questionnaire including questions on their perception of finding errors in StoryTec-authored games without and with model checking and a question whether this functionality should be included by default.

Table 8 provides an overview of the amount of errors found without and with the model checking functionality. Note that the participants continued with the same game, therefore, the remaining number of errors to be found was determined by the number of errors found during the first round of the experiment. Only one participant was able to find all errors without assistance. In the second round, all other participants were able to find all remaining errors. By comparing the times spent in each round with the number of errors found, we see that on average, participants required 3.9 minutes to find an error during the first phase (without model checking), and only 1.96 minutes during the second phase (with model checking).

Table 9 shows the number of times each of the errors included in the game have been found without assistance. We find that while 5 out of 6 participants were able to find the malformed action error without assistance, only 2 out of 6 could find the unreachable scene. This clearly motivates the need for model checking, since even for this confined example, some logical errors can be hard to track.

This finding is corroborated by the results of the questionnaire shown in Figure 42. Each questionnaire item was a statement which participants rated from 0 (not

|                              |              | PARTICIPANT |     |     |     |     |     | AVERAGE |
|------------------------------|--------------|-------------|-----|-----|-----|-----|-----|---------|
|                              |              | 1           | 2   | 3   | 4   | 5   | 6   |         |
| WITHOUT<br>MODEL<br>CHECKING | TIME         | 7           | 13  | 11  | 11  | 11  | 12  | 10.83   |
|                              | ERRORS FOUND | 4           | 5   | 3   | 0   | 2   | 2   | 2.67    |
|                              | TIME/ERROR   | 1.75        | 2.6 | 3.7 |     | 5.5 | 6   | 3.9     |
| WITH<br>MODEL<br>CHECKING    | TIME         | 1           | 0   | 4   | 14  | 5   | 7   | 5.17    |
|                              | ERRORS FOUND | 1           | 0   | 2   | 5   | 3   | 3   | 2.33    |
|                              | TIME/ERROR   | 1           |     | 2   | 2.8 | 1.7 | 2.3 | 1.96    |
| IMPROVEMENT                  |              | 0.25        |     | 0.7 |     | 0.8 | 1.2 | 0.73    |

Table 8: Performance data from the pilot study on model checking.

| ERROR               | PARTICIPANT |   |   |   |   |   | TIMES FOUND | AVERAGE |
|---------------------|-------------|---|---|---|---|---|-------------|---------|
|                     | 1           | 2 | 3 | 4 | 5 | 6 |             |         |
| UNREACHABLE SCENE   |             | 1 | 1 |   |   |   | 2           | 33%     |
| MALFORMED ACTION    | 1           | 1 | 1 |   | 1 | 1 | 5           | 83%     |
| MALFORMED CONDITION | 1           | 1 |   |   | 1 |   | 3           | 56%     |
| UNUSED TRANSITION   | 1           | 1 |   |   |   |   | 2           | 33%     |
| UNUSED VARIABLE     | 1           | 1 | 1 |   |   | 1 | 4           | 67%     |

Table 9: Participants of the model checking study and absolute and relative number of times each error type has been found.

applicable) to 4 (fully applicable). The participants rated the error-finding procedure without the assistance with only 1.17 in average, whereas the usefulness of the model checking functionality was rated with 3.5.

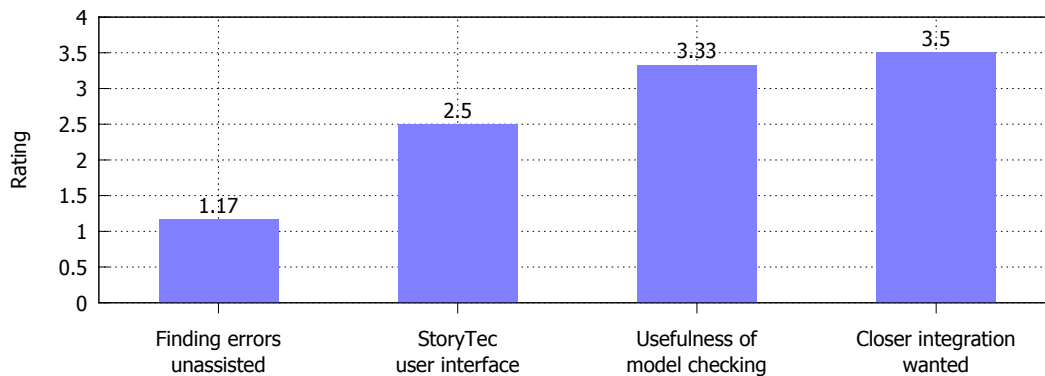


Figure 42: The results of the questionnaire of the model checking experiment (the range of values is 0 to 4.)

While this first experiment shows the applicability of model checking, the small sample size limits the weight that can be given to it. We see it as an indication that model checking is a useful addition to an authoring tool and will add more intricate queries to the system in the future.

## 6.5 CONCLUSION

This chapter represented the link between the abstract model elaborated to capture and transcribe a game on the one hand and the author using an authoring tool on the other side. Therefore, this chapter focused on the way in which an author could interact with the model of the game.

The major conceptual results of this chapter are the user support mechanisms of template-based authoring, collaborative authoring, iterative authoring and model checking.

The field of template-based authoring was subdivided into structural templates and interaction templates. Structural templates allow the definition of re-usable game parts which can quickly be instantiated during authoring in order to fill a game structure quickly or to provide a best practice guideline for a game structure. Interaction templates complement this structural approach by encapsulating complex game mechanics and thereby realizing a separation of concerns between programmers, who provide the implementation of an interaction template, and authors, who configure only the relevant properties of interaction templates. As shown in chapter 8, interaction templates can increase the efficiency and effectiveness of an authoring tool significantly by providing game mechanics that would be hard or impossible to model using only visual programming approaches.

For collaborative authoring, we described the concepts of direct and indirect collaboration. In both collaboration schemes, we have shown the game model described in chapter 4 to be a foundation for collaborative work. In direct collaboration, where a team works on a game simultaneously, the integrated nature of the authoring tool



concept, including all game development roles, allows all involved persons to see and work on the game. In indirect collaboration, where one author creates a game and a second author customizes it, an extension to the game model establishing accessibility levels allows the authoring tool to only display the relevant information to the customizing author.

The concept for a rapid prototyping and analysis tool which is intended to aid in the iterative development of games was presented. This tool allows game to be tested at an early stage, yielding feedback for authors concerning parameters such as the effects of adaptive algorithms on the game.

Finally, model checking as a method for identifying potential problems in a game that cannot be prevented by enforcing the structures of the game model. Hereby, a model checker can identify conditions such as a variable that has not been used or a dead-end which interferes with a player successfully finishing the game and warn the author. An initial small-scale study has been carried out, identifying both a performance increase in authors searching for semantic errors in a game and a wish by authors for model checking to be included in a game authoring tool.

The implementation of the authoring tool concept including the user support mechanisms as major components is presented in the following chapter.

## IMPLEMENTATION

IN this chapter, we present the main result of this thesis, the implementation of the concept detailed in the previous chapters in the form of the authoring tool StoryTec. We discuss the implementation details of adaptive features as described in chapter 5 as well as the implementation of interaction templates as described in chapter 6. This discussion is followed by a presentation of a mapping of the educational game development roles introduced in chapter 2 into the components of the authoring tool. Following this presentation, we describe a basic architecture for a run-time environment for adaptive educational games which is used to execute games authored with StoryTec. Finally, the implementation of the rapid prototyping and scientific analysis tool StoryPlay is provided.

## 7.1 AUTHORING TOOL IMPLEMENTATION: STORYTEC

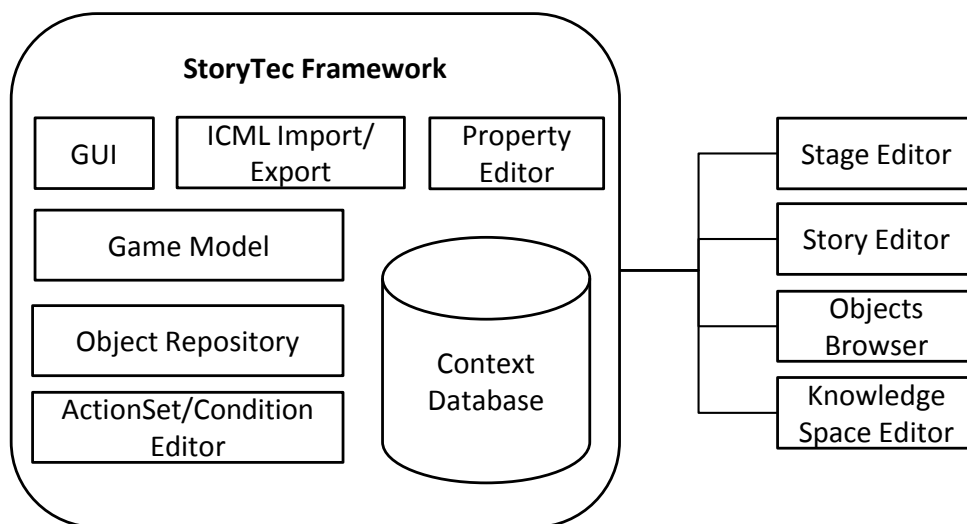


Figure 43: The architecture of StoryTec.

The architecture of StoryTec is based upon a core framework and several plugins, as can be seen in figure 43. The core framework contains the implementation of the game model and the necessary code for providing a user interface to the model. Specialized tasks are given to the plugins for the Story Editor, Stage Editor, Objects Browser and Knowledge Space Editor as described below.

The main user interface consists of the Story Editor, Stage Editor, Objects Browser and Property Editor, as can be seen in figure 44<sup>1</sup>.

<sup>1</sup> Screenshots in this section are created with content from the game “Favourite Places” created as a tutorial for StoryTec available to the open community as described in appendix A.1



Figure 44: The default user interface of StoryTec. Individual components will be shown in separate figures for better readability in print; this screenshot serves as an overview.

### 7.1.1 Story Editor

The Story Editor is the main interface to the overall model of the game that is worked on in StoryTec. It visualizes scenes, objects as well as transitions and allows these elements to be edited. Users can zoom freely in order to get an overview of the whole game structure or to focus on individual parts. Figure 45 is a screenshot of the Story Editor with an overview of the example game.

Scenes are visualized as boxes that can be freely moved and resized. A special symbol indicates that the game should start with a given scene. Objects are shown as smaller, non-resizable rectangles along with the icon of the object type inside their scenes and a preview of their content as a semi-transparent background. Therefore, authors get an abstract overview of the whole story in the Story Editor. For editing details, the Stage Editor is provided.

Transitions between scenes can be drawn in the user interface and are visualized as arrows, pointing from the scene the transition leads from to the target scene. A transition can be switched between being fixed (triggered explicitly) and free (triggered from an adaptive scene change). This difference is also visualized.

Compared to other tools featuring the concept of a Story Editor (such as e-Adventure, see section 2.3.3), StoryTec's implementation of this editor allows the free editing of all visualized entities and is not restricted to mainly visualization.

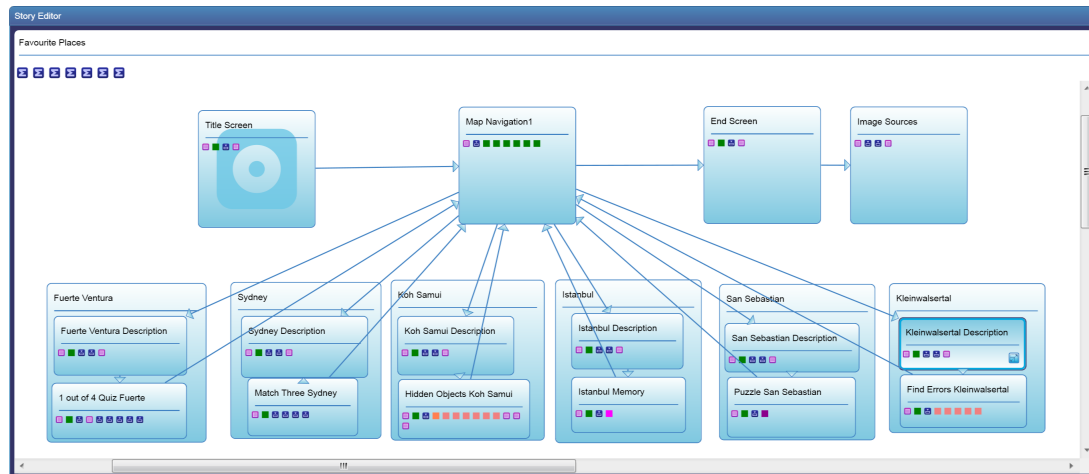


Figure 45: The Story Editor for manipulating the game's structure.

### 7.1.2 Stage Editor

The Stage Editor shows the details of the scene currently selected in the Story Editor and allows editing the objects found in the scene. Instead of an abstract view in the Story Editor, this visualization is carried out in a fashion maximally true to the look the scene will have in the game (What You see is what you get - [WYSIWYG](#)). Figure is a screenshot of the Stage Editor for a scene in the “Favourite Places” example with a “memory” interaction template. Authors can specify the images that are to be matched and the text to match them with directly in the Stage Editor without switching to another editor.

For 2D games, this can be realized by re-creating the same look in the Stage Editor as the game will later have. For 3D games, a compromise between true 3D editing and usability for non-specialists has to be found. Two approaches to this compromise have been undertaken in the course of this thesis. For the 3D educational game realized in the 80Days project (see appendix [A.2](#) for details), an intuitive abstraction of the 2D world is a map on which authors can place game tasks. While this does not re-create the actual 3D world of the game, it allows authors to work on the game without understanding the basics of navigation and editing in a 3D environment. A second realization is the integration of the Techcraft engine for block-based games similar to Minecraft (cf. section [2.1](#)). In this Stage Editor, authors can jump into a game-like environment which uses simple controls similar to those of Minecraft and allows authors without training in 3D content creation software to create 3D terrains and objects. Details can be found in appendix [A.8](#).

The Stage Editor relies on the authoring implementation of interaction templates for visualizing details of an interaction template. The individual interaction templates are therefore integrated into the frame offered by the Stage Editor. Dragging an object onto the Stage Editor from the Objects Browser will instantiate it, if possible at the position where it was dropped.



Figure 46: The Stage Editor, providing WYSIWYG editing.

### 7.1.3 Objects Browser

The objects browser provides authors with a list of all instantiatable objects, as well as of both structural and interaction templates. By double-clicking or by dragging and dropping, authors can instantiate the available objects in the currently selected scene. Object prototypes are sorted according to their type. If the object is dragged to the Stage Editor, it is instantiated in the currently selected scene. If it is dragged to a scene in the Story Editor, it is instantiated in this scene. By using these mechanisms, authors can either quickly add a set of objects to several scenes to fill the game with content quickly or concentrate on one scene at a time.

Note that there are two kinds of object prototypes that are offered in the Objects Browser. The types are characterized by them being pre-configured with certain data or lacking this pre-configuration. In certain cases, it is natural to provide such pre-configured objects. For example, they are used in the case of the “Mark difference” interaction template, where differently shaped clickable areas are used to mark differences between two images. The different shapes are realized by different pre-configured instances of the same type of object. On the other hand, especially objects such as clickable hotspots which do not have a visual look by themselves should be offered as freely configurable objects.

### 7.1.4 Property Editor

The Property Editor is the authors’ main interface to parameters of objects and scenes. According to the type of the currently selected item, it dynamically creates the necessary input fields for editing properties. Apart from standard editing widgets such as checkboxes for Boolean values, it also provides editing capabilities for metadata

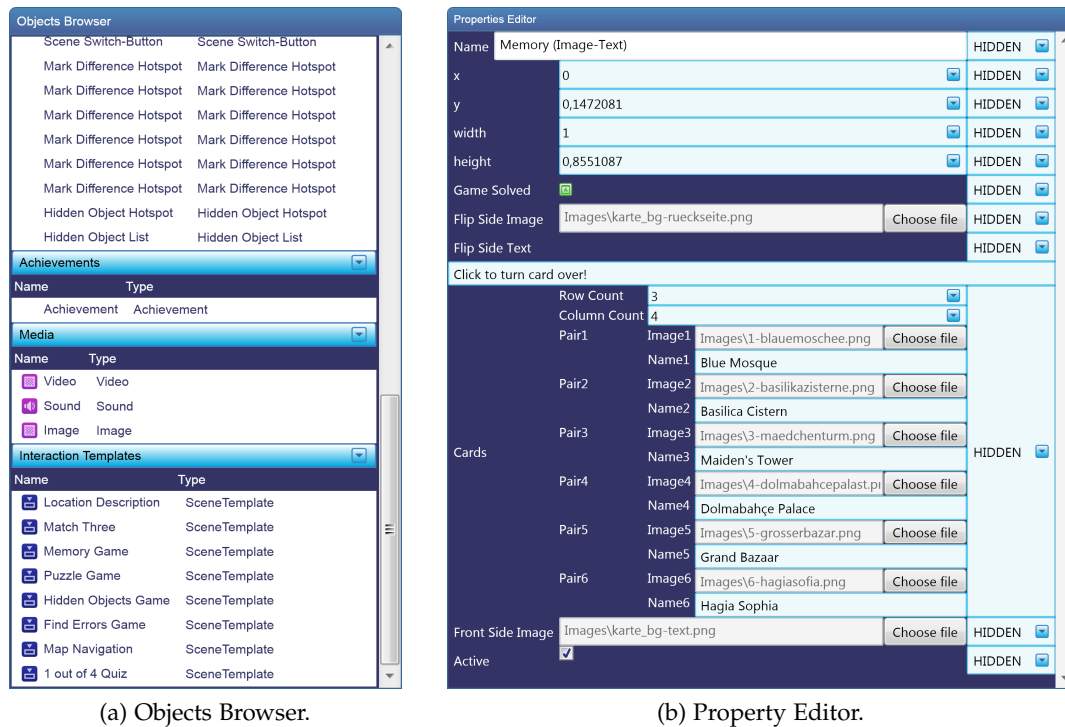


Figure 47: The Objects Browser and the Property Editor.

such as a set of sliders for the appropriateness of scenes concerning narrative, learning and playing for NGLOBs.

#### 7.1.5 Skill Tree Editor

Since skills to be learned in StoryTec games are organized according to Competency-based Knowledge Space Theory (see chapter 5), an editor allowing such structures to be edited visually is included in StoryTec. Using this Knowledge Space Editor, authors can create skills which are then visualized as boxes. It provides fields for assigning an identifier to the skill along with a description of the contents related to this skill. Once several skills have been added, authors can draw connections between skills using the mouse cursor. A connection from one skill to another skill corresponds to a prerequisite relationship between the skills.

In this way, a logical structure of skills results from the editing process, which is used by the adaptive algorithms in the Story Engine on the one hand and which can help authors with retaining an overview of the game's curriculum during authoring on the other hand.

#### 7.1.6 ActionSet and Condition Editor

Authors can open the ActionSet editor whenever a stimulus is visible in the authoring tool, for example among the parameters of an interaction template. It interactively visualizes the graph structure of actions established in chapter 4. Actions are shown as boxes whose visual alignment is mainly vertical, similar to a flow diagram.

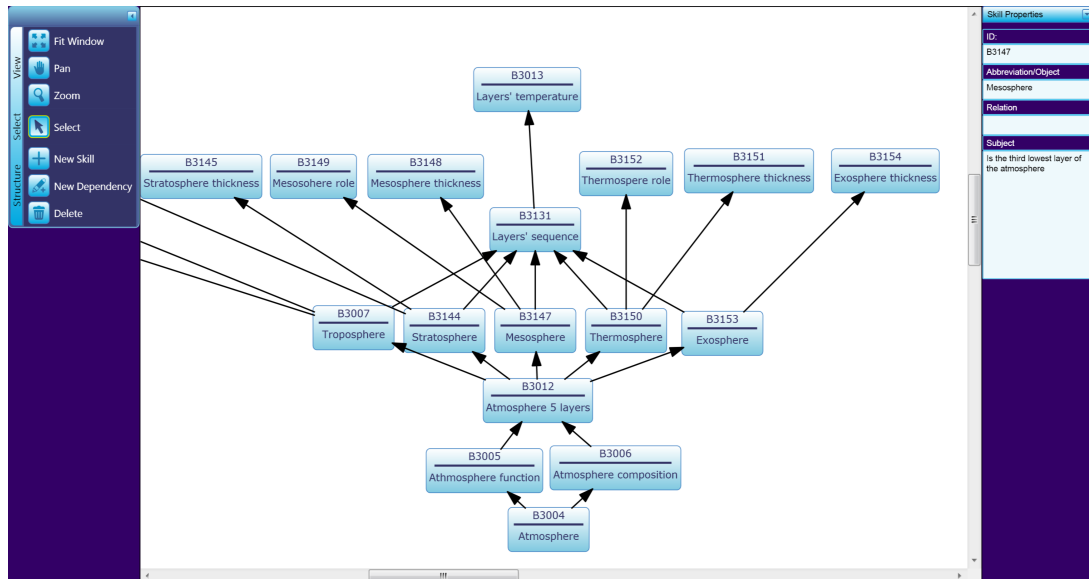


Figure 48: The Skill Tree Editor.

Actions that are visualized below other actions are executed after the previous one above it has been executed. Arrows help in reading this diagram. Branches based on conditions are visualized by arranging the two or more resulting strands of execution horizontally on the same level.

Authors can create new actions at any point in the existing graph structure. After an action has been created, it is empty at first. In order to assign the correct action, the author is lead to a valid action by a three-step process. First, he or she is asked for the type of object the action is to be executed upon. In this as well as the following step, only those objects and object types which are actually relevant by being in the scope of the current action set are shown. After the object type has been decided, the second step is to choose the object to execute the action on. If this object only has one type of action, it is chosen automatically. Otherwise, the author has to choose the action manually. After a concrete action has been chosen, editing widgets for all parameters of this action are created dynamically as in the Property Editor. Using these forms, authors can edit the parameters of the action.

This process aids authors in that it only presents relevant objects which are in scope and by leading them from a broad choice of object type to the concrete action in question instead of providing all actions simultaneously.

At any position in the graph, authors can create conditions. These conditions, determine at runtime whether the sequence of actions below them is to be executed. Clicking on a condition opens the Condition Editor. This editor allows editing of Boolean expressions in a visual way. A single Boolean expression is used to compare a value (for example the current value of a variable at runtime ) with another value. Several expressions can be combined to form a term. For each expression, the author can choose how it is combined with the next expression by the usual Boolean operators AND and OR. In order to allow bracketing, the Condition Editor allows single expressions or sets of expressions to be indented, indicating that the indented expressions are at the same level of bracketing.



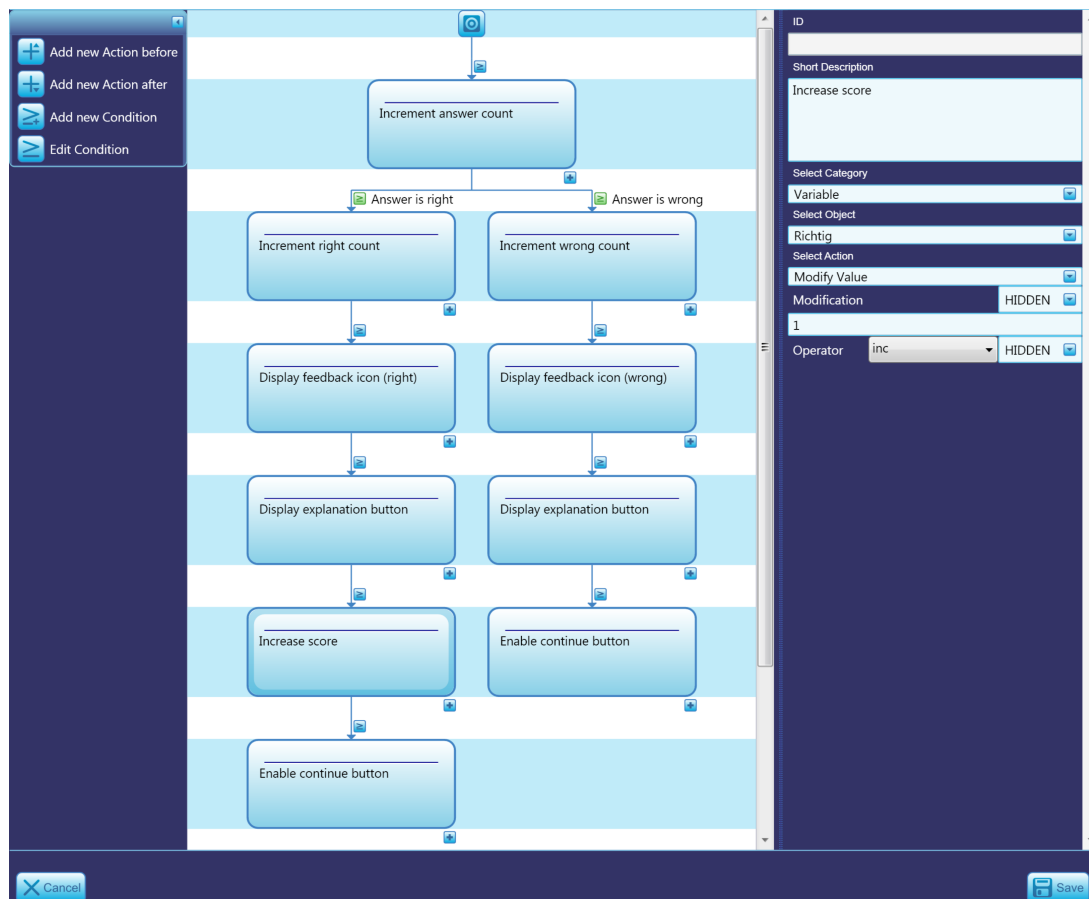


Figure 49: The ActionSet Editor.

Both actions and conditions can have a description which is visualized along with the graph structure. For both, a default is generated from the structure of the action or condition. This default can be overridden in case it is too long or if an author prefers another way of documenting the action or condition.

## 7.2 IMPLEMENTATION OF ADAPTIVE FEATURES

The features for authoring of adaptive games are realized in several areas of the authoring tool and therefore available to all authors. The first component involved is the Story Editor in which the overall game structure and all possible paths through the game are visible. The adaptive parts of the game can be seen since scene pools and free transitions are visualized differently than other structural elements. This aids authors in keeping an overview of the ways in which the game can be adapted.

As noted previously, making a game adaptive requires the creation of alternative paths through the game, often with different content or gameplay. This overhead is mitigated by allowing authors to create these new paths quickly, populating them with gameplay by using interaction templates and furthermore by offering the functionality to copy and then vary an already existing structure. This can be achieved either by the copy & paste-functionality for limited and local re-use on the one hand and by the structure template mechanism for often-reused structures on the other hand.



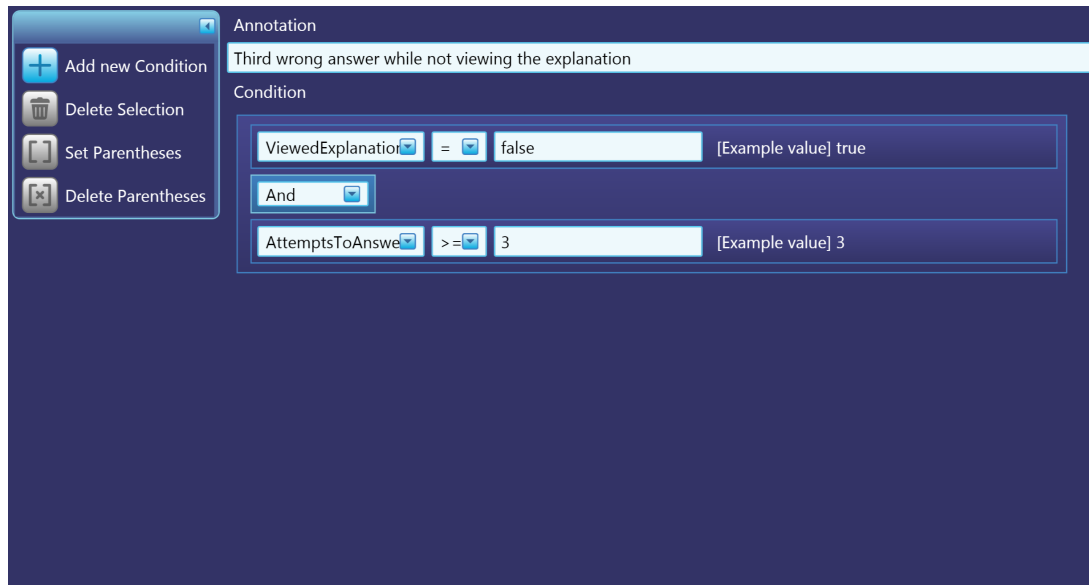


Figure 50: The Condition Editor.

The last component aiding in creating adaptive games is the rapid prototyping tool StoryPlay, in which authors, after creating the structure for the game with adaptable paths, can quickly test their game and the effects of the adaptive algorithms on it. This allows them to find out, using early prototypes, how the game reacts to different inputs during play sessions.

The implementation of the adaptive mechanisms of NGLOBs is embedded in the user interface of StoryTec and available from several contexts:

- The structures of player and narrative models are loaded at startup. As described above, the categories by Bartle respectively the steps of the Hero's Journey are supplied as the default models; however, since they are not hard-coded into the tool, they are replaceable by other models if this required.
- In order to define the curriculum of the game, the Skill Tree Editor is provided. It offers an interface to visually create a graph structure, which defines the competencies (nodes) and dependencies (edges).
- In order to annotate scenes and thereby elevate them to NGLOBs, the Property Editor has dedicated sections in which the necessary data can be entered. For the player and narrative model, input widgets are created which allow the author to specify values in the range [0, 1] for the categories/steps. In order to annotate scenes with prerequisite and associated skills, the author uses the skills defined in the Skill Tree Editor similar to tags.
- To enable the adaptive mechanism, authors can either mark transitions as free in the Story Editor or place the scenes into scene pools using the Story Editor.
- Finally, the model updates are integrated into the system of stimuli in the ActionSet Editor. Since each relevant action by the player triggers a stimulus, it makes sense to add the model updates as actions. This is realized by actions for learner and player model updates, where authors can specify "new" values for

these models which are then combined with the previously observed values at runtime as described in chapter 5.

The information about NGLOBs is saved along with the remaining game data in the ICML file format as described above. This data is loaded by the re-usable Story Engine component which controls a player application or game engine.

The Story Engine contains the encoded player, learner and narrative models and reacts directly to update-actions by updating the respective model. Whenever an adaptive choice is called for, the system relies on a set of priorities that assign weights to the three NGLOB axes.

### 7.3 IMPLEMENTATION OF INTERACTION TEMPLATES

As described in chapter 6, interaction templates are used to allow authors to configure complex game mechanics without having to program. The authoring tool implementation of interaction templates is placed in the Stage Editor when available and helps authors by offering WYSIWYG editing of the interaction template.

The authoring tool implementation of an interaction template can also assist the authoring framework in providing a wizard that helps authors in correctly configuring the interaction template. Figures 51 to 54 show the wizard that is created for configuring an example interaction template which was introduced in chapter 6, in which a set of letters has to be arranged to form the correct codeword. After opening the wizard, the author is first asked to choose an interaction template to use (see Figure 51). A list of available interaction templates is presented.

Associated with each template is a set of automatically (by utilizing the interface definition) and manually generated documentation, which is presented in a summary page. This page can include a description of the interaction template specifying what authors can expect from the template, a video preview of a reference implementation, as well as the optional and required assets and parameters that authors are asked to provide during the wizard process. Figure 51 shows this overview page.

A second wizard page is reached when an interaction template has been chosen. This page and following pages lead the author through the configuration of the template. Figure 52 displays this wizard page, in which a text field is created that accepts the correct codeword. As indicated in the diagram, this page can be generated by analyzing the interface definition of the template and recognizing that a string parameter is required. The associated text that explains the purpose of this parameter can be drawn from the manual template documentation, if available.

Figure 53 displays a similar wizard page for the optional parameter of a background image to place in the background of the screen while the interaction template is active. Again, the explanation text is drawn from the associated documentation.

After the choice of an interaction template in the first wizard screen and the subsequent configuration of parameters in the following two screens, the final screen is generated from the definition of stimuli that the interaction template can trigger (see Figure 54). For the example interaction template, we have defined one stimulus that is triggered when the game is successfully completed. In order to configure the resulting actions, a button is provided that opens the ActionSet Editor, the component that allows authors to program using the visual programming language in

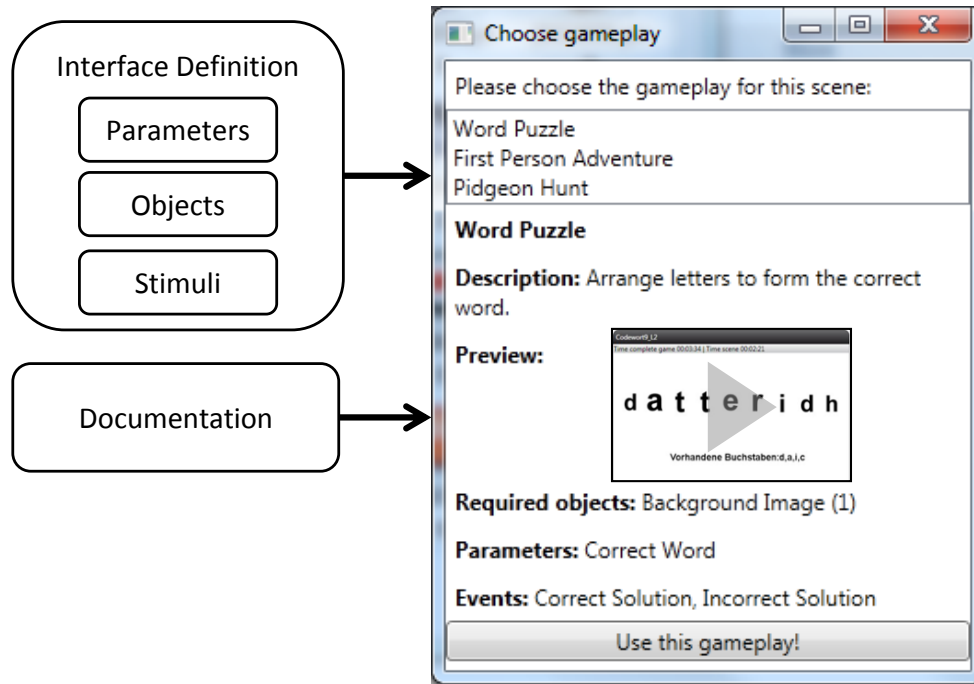


Figure 51: The initial state of the wizard, in which the interaction template is chosen.

StoryTec. Possible actions include congratulating the player in text form, playing a sound, triggering a transition to change the scene or any of the other options found in StoryTec.

As noted in chapter 6, two advantages of interaction templates are the separation of concerns concerning programming and authoring and the portability afforded by the concept. We see the first advantage in the example interaction template: all implementation issues are handled in the authoring tool during exporting and in the player application at runtime. Implementing the game mechanics from scratch would require assembling images of all letters, writing the code for displaying and cycling the letters and checking the correctness of the solution. The second advantage is demonstrated in figure 55. A 2D implementation of this word-puzzle interaction template was shown in figure 32. Figure 55 shows a version adapted for 3D, in which the letters are placed on polyhedra aligned on an axis. Adjusting to the affordances of 3D, players can now see the other letters on the sides of the polyhedra.

Even though the implementations are quite different concerning presentation and user interaction, both are configured from the same settings an author made in StoryTec. All adaptations are carried out automatically without any later interventions by authors.

#### 7.4 AUTHORING PROCESS

Figure 56 shows a subset of the most relevant mappings from game development roles to parts of StoryTec and the associated editors and applications. The different user groups are mainly separated by the way in which their work is carried out directly in the authoring tool or how it fits in the resulting workflow.

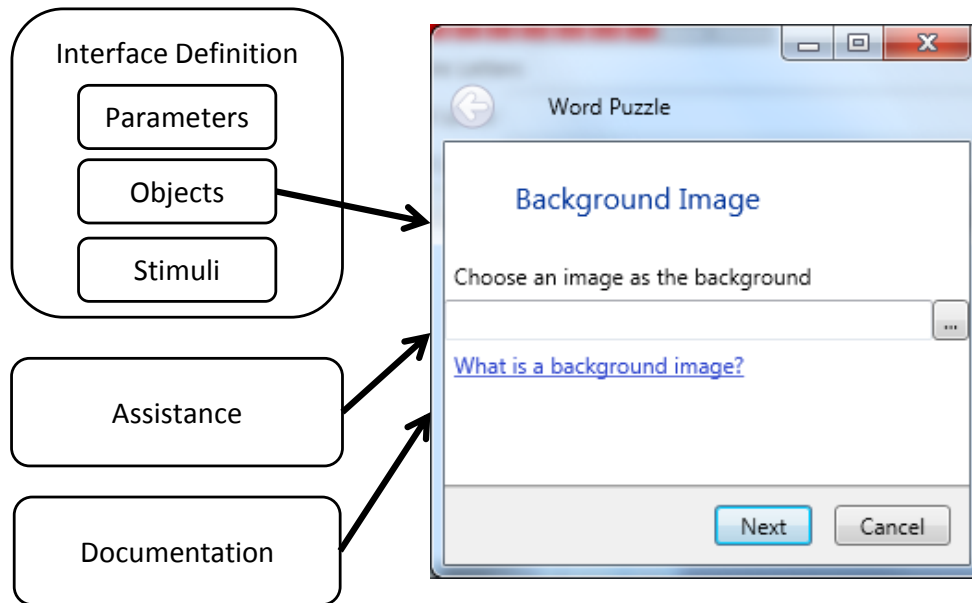


Figure 52: The second wizard page, in which the correct solution is queried.

Technicians (programmers) are mainly involved in supplying the implementations of interaction templates that can be included in games and re-used whenever appropriate in game production. In carrying out this task, they are assisted by the component-based approach of interaction templates. We can further subdivide this group into authoring tool programmers which provide the authoring tool implementation and game programmers which provide the runtime implementation in a game engine. The definition and design of the templates is carried out beforehand by game designers. The programmers also provide the basic framework for running the game, i.e. the game engine.

The work of the remaining user groups is mainly done in the frontend user interface of the authoring tool. By being able to see and control various aspects of the game in an interface shared by all authors, they are able to communicate and collaborate about the tasks coming up in the game production process.

Authors take part in a workflow in which they choose the interaction templates to use and then configure the parameters that are relevant to them. Apart from textual content, this can include assets created by artists. It can be noted that this workflow gives authors a certain structured design space, spanned by the modeling capabilities of the game model and the features of the interaction templates. By limiting this design space, we allow a separation of concerns between authors and programmers, with the former not having to implement actual game mechanics, but instead working with configurable implementations. “Authors” is here used as the umbrella term for all users working with the authoring tool frontend. Expert authoring tool users are those that are able to work with the complete authoring tool and configure the domain-specific views for domain experts, specifying the parts of the game and options visible to them.

Game designers can use the interface of StoryTec to structure the game using the Story Editor and define the narrative and the locations at which it takes place. By using the visual programming approach of the Action Set Editor, they are empowered to manage the high-level flow of the game without having to be aware of implemen-

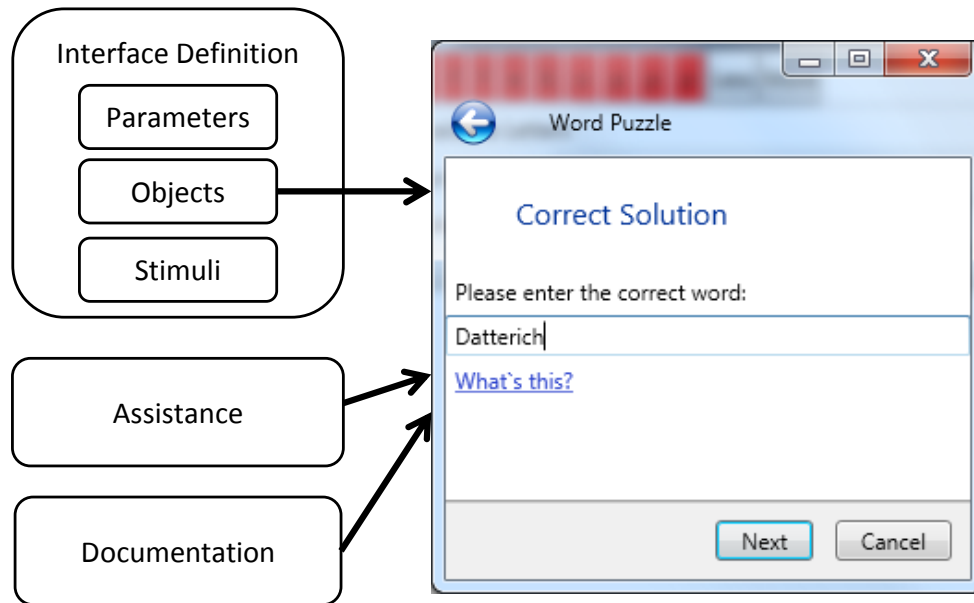


Figure 53: The third wizard page, querying the background image for the game.

tation details. In an initial step, they define the designs and specifications for the configurable and re-usable interaction templates to be implemented by technicians.

Domain experts are able to enter the curriculum of skills to be taught in the game using the Knowledge Space Editor. They can fill the game with content according to their expertise, especially by making use of the collaborative authoring functionality, which only presents them with options that are designated to them. An example implementation of this feature can be found in the Motivation 60+ project, described in Appendix A.10.

## 7.5 RUNTIME ENVIRONMENT ARCHITECTURE

The basic architecture of applications authored with StoryTec is shown in figure 57. As can be seen, the central component in this architecture is the Story Engine. This software component is configured by the project files written from StoryTec. The Story Engine communicates with the application or game engine via an interface that is based on the primitives defined for the game model in chapter 4. Stimuli are used to send relevant game events from the game to the Story Engine. Information about objects is sent via parameters to the game to initialize them. Actions are sent from the Story Engine to the game, changing the state of the game and initiating game events such as dialogues with characters or scene changes. Information about NGLOBs is used in the game to load the appropriate environment and settings.

The Story Engine acts as a layer of abstraction above the target system (the actual game engine running the game). The game engine is controlled by high-level commands which are then broken down to the level of the game engine. This process is mainly carried out in the runtime implementations of interaction templates. In essence, this leads to the technical development being clearly carried out in the game engine, while the content production for the game culminates in the authoring tool, where assets created with various digital content creation tools are aggregated.

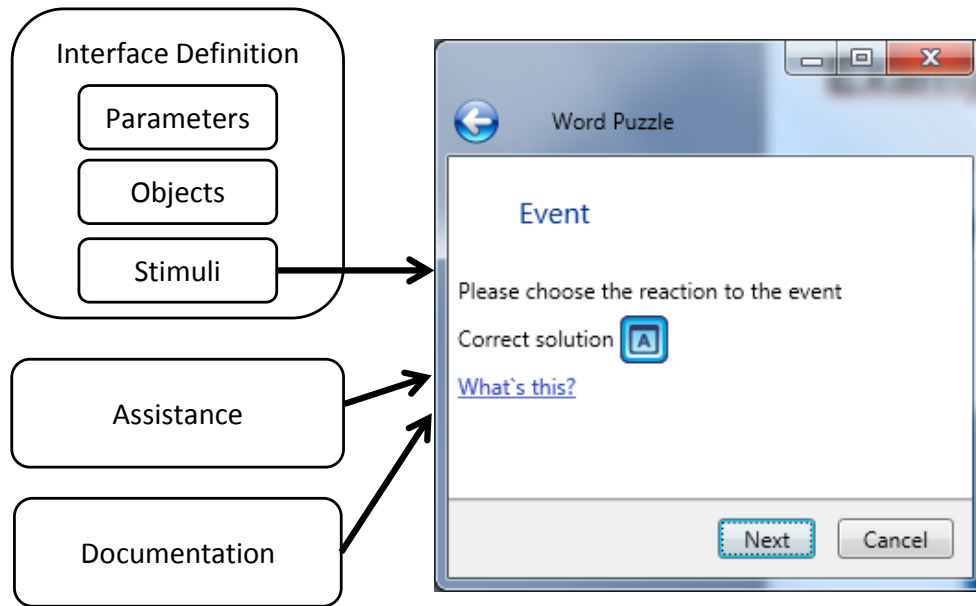


Figure 54: The final wizard page, where the stimulus for the successful solution of the game is offered to be edited.

The game includes the necessary mechanisms to be connected to the Story Engine. These mechanisms are not prescribed in the architecture and are determined by the target platform and its properties. They have been realized in various contexts. One option is embedding the Story Engine directly into the game code, allowing the communication overhead and latency being minimized. However, this requires the Story Engine to be written in the programming language of the game, limiting the applicability of this solution. Another option implemented during this thesis is the communication via local TCP/IP connections or via websockets in the case of an HTML 5 version. This connection option minimizes the integration work and allows one version of the Story Engine to be connected to many different games, however, it introduces overhead and latency, especially if the Story Engine is placed on a server connected to the game via the internet.

Apart from controlling the game, the Story Engine it is responsible for managing the adaptive story, player and learner models in order to realize adaptive games. For this purpose, it can optionally be connected to a dedicated learning engine that enhances the accuracy of the learner model. This optional part of the architecture was used in the 80Days project and is detailed in appendix A.2.

The Story Engine makes use of an xml-based exchange format called ICML<sup>2</sup>. This format was originally defined as the description language for interactive stories in the context of the previously mentioned project INSCAPE. The original version of this language was completely overworked in order to allow novel concepts, including interaction templates and support for the adaptive features of NGLOBs.

Figure 58 provides an overview of the main elements of an ICML file. As can be seen, the main element for structuring the game are scenes, which represent one scene, potentially annotated with adaptive information to become an NGLOB. One of the major revisions made to ICML since the use in INSCAPE is the omission of

<sup>2</sup> INSCAPE communication markup language



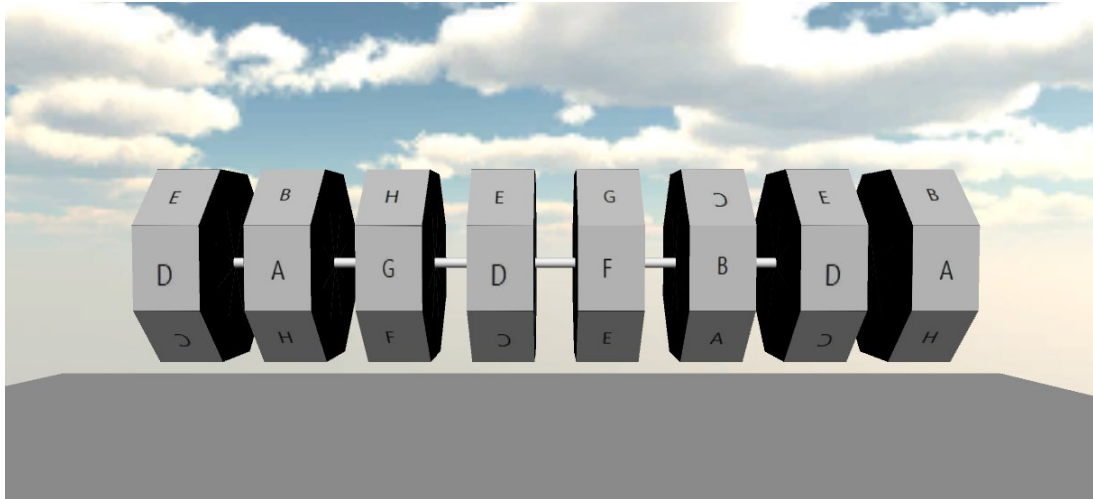


Figure 55: The example interaction template realized in 3D using the Unity3D game engine.

the type `complexScene` that was found the initial usability study described in the following chapter to be confusing for authors. A `complexScene` was used for hierarchical organization of scenes, with scene elements being only used for the leaves of the game graph. Since this lead to confusion in the usability study participants, the only remaining type of

Table 10 provides an overview how the concepts defined in chapters 4 to 6 have been mapped to elements of ICML.

ICML-encoded games are read into the Story Engine which interprets the files and controls the game engine based on the configuration by the author in StoryTec. scene organization is the type `scene`.

Figure 59 shows the xml schema that is used to save and exchange information about the three contexts of a NGLOB. A context is defined for each adaptive axis, resulting in the `narrativeContext`, `learningContext` and `gamingContext` elements. Apart from a textual description intended for note-taking of authors or collaboration between different authors, specialized elements are used to capture the quantifiable aspect of NGLOBs. For the narrative context, `dramaturgyAppropriateness` elements are used to encode the stages of a story model the NGLOB addresses and the percentage to which it does so. Lists of prerequisite and appropriate skills are defined for the `learningContext`. The `playerModelAppropriateness` elements are used to quantify the percentage to which a NGLOB is suited for one of the player types of the game.

The encoding of conditions and actions is shown in the schema for action sets as shown in figure 60. An action element targets an object or a transition and specifies which action is intended to be executed by the game engine along with the parameters of the action (for example, the line of dialogue a character should say). Both condition and stimulus elements can be used in `actionSets`. The child elements of a condition are only executed if the condition is true, while the child elements of a stimulus are only executed when the game has interpreted a game event to correspond to the stimulus and triggered it.

The Story Engine component is created as a modular component that can be introduced to various game engines. In the course of the development of StoryTec,

| CONCEPT                 | REPRESENTATION IN ICML  |
|-------------------------|---|
| GAME MODEL              |   |
| Scene                   | scene, hierarchical organization possible   |
| Object                  | object inside scene's stageSet  |
| Transition              | transition inside origin scene's transitionSet  |
| Parameter               | param with multiple data types (boolean, string, integer, decimal, float)               |
| Action                  | action tag in scene's actionSet   |
| Condition               | condition in scene's actionSet  |
| Stimulus                | stimulus in scene's actionSet   |
| ADAPTIVITY              |   |
| Narrative Model         | Annotation in scene's narrativeContext  |
| Player Model            | Annotation in scene's gamingContext   |
| Learning Model          | Annotation in scene's learningContext   |
| Free Transition         | Marked by setting transition attribute free to true                                     |
| Updates                 | Realized as special actions   |
| AUTHOR SUPPORT          |   |
| Structural Templates    | Realized by pre-configured ICML-files included in the authoring tool                    |
| Interaction Templates   | Realized by special objects   |
| Collaborative Authoring | Realized by access control parameters for ICML elements (scene, transition, param, ...) |
| Iterative Authoring     | Realized outside of ICML  |
| Model Checking          | Realized outside of ICML  |

Table 10: An overview of the implementation of concepts as ICML elements.



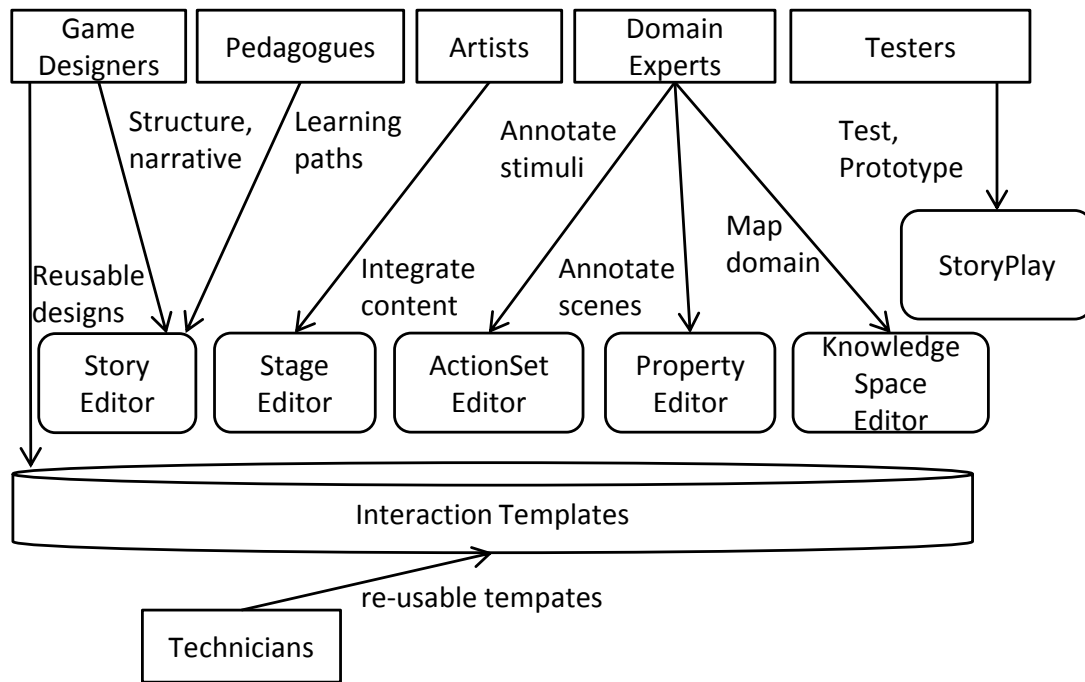


Figure 56: Some of the possible mappings between user groups and the components of the described authoring tool. Note that not all combinations are shown, for example, game designers could also use StoryTec and StoryPlay in conjunction for storyboarding and rapid prototyping.

this modular concept has been demonstrated with several game engines. Examples of the Story Engine controlling the flow of games on a high level of abstraction are provided in appendix A.

As described in chapter 6, the component-based concept behind interaction templates assists with cross-platform publishing by supplying developers with clear definitions of involved objects and intended game mechanics. As was noted there, the main work concerning portability lies therefore in creating player implementations for each target platform. In our work, this has further been mitigated by utilizing a cross-platform programming language (Haxe<sup>3</sup>), along with a 2D game engine written for this programming language (Kha, by KTX Software Development<sup>4</sup>).

Interaction template implementations are written in Haxe and implemented in the Kha engine, drawing upon an API which is implemented platform-independently. The only code that has to be adapted to a new target platform is localized in specialized backends which call platform-specific code to realize engine functionality. Therefore, the player implementations of interaction templates are already platform-independent in this implementation scheme.

Figure 61 visualizes the process of creating versions of the player for new target platforms by utilizing a core of features (including the Story Engine as well as interaction template implementations) programmed platform-independently in Haxe and implementing only a set of platform-specific functions which are mainly related to establish the connection to the graphics device of the target platform.

<sup>3</sup> <http://haxe.org/>, last visited on January 24, 2013

<sup>4</sup> <http://www.ktx-software.com/>, last visited on January 24, 2013

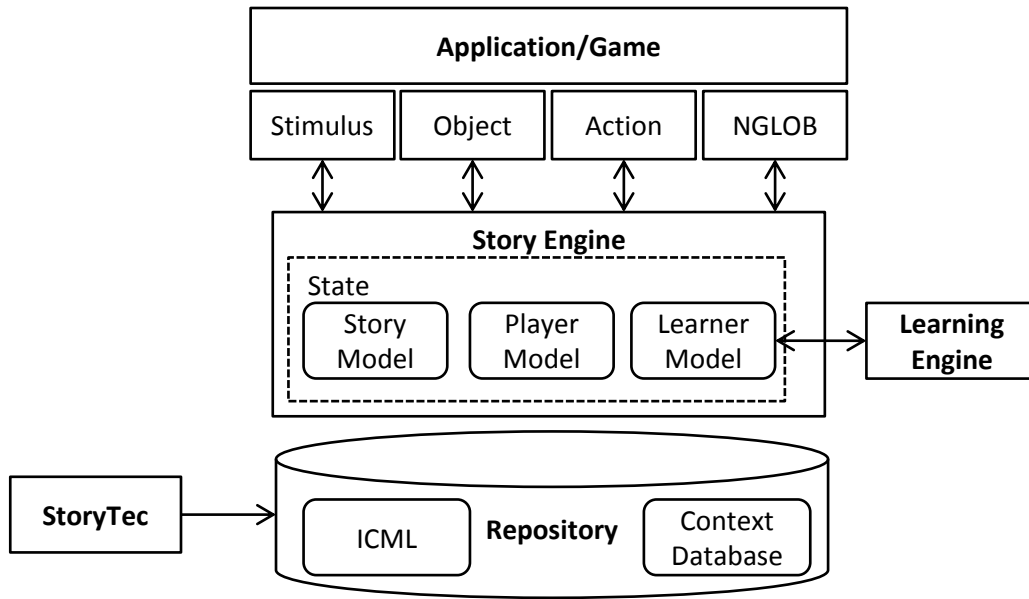


Figure 57: The architecture of games authored with StoryTec.

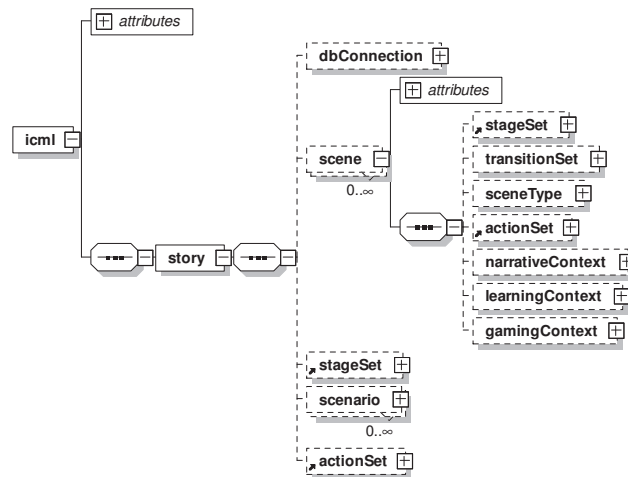


Figure 58: The overview of a game model encoded in ICML.

## 7.6 RAPID PROTOTYPING TOOL IMPLEMENTATION: STORYPLAY

Apart from the StoryTec authoring tool, the suite of tools that comprise the StoryTec environment also contains two player applications. The cross-platform player application “StoryPublish” is intended to be used by end-users and exposes only an interface to the game itself.

The concept of a rapid prototyping and scientific analysis tool as described in chapter 6 is realized in the “StoryPlay” player application. It features an interface separated into two parts (cf. figure 62). The first part realizes the game and implements interaction templates, and in later versions is implemented by embedding the actual “StoryPublish” player to be as authentic as possible. The second part is used to visualize the relevant data concerning the history of choices in the game and evolution of the NGLOB models. As described in chapter 6, it can therefore be used by

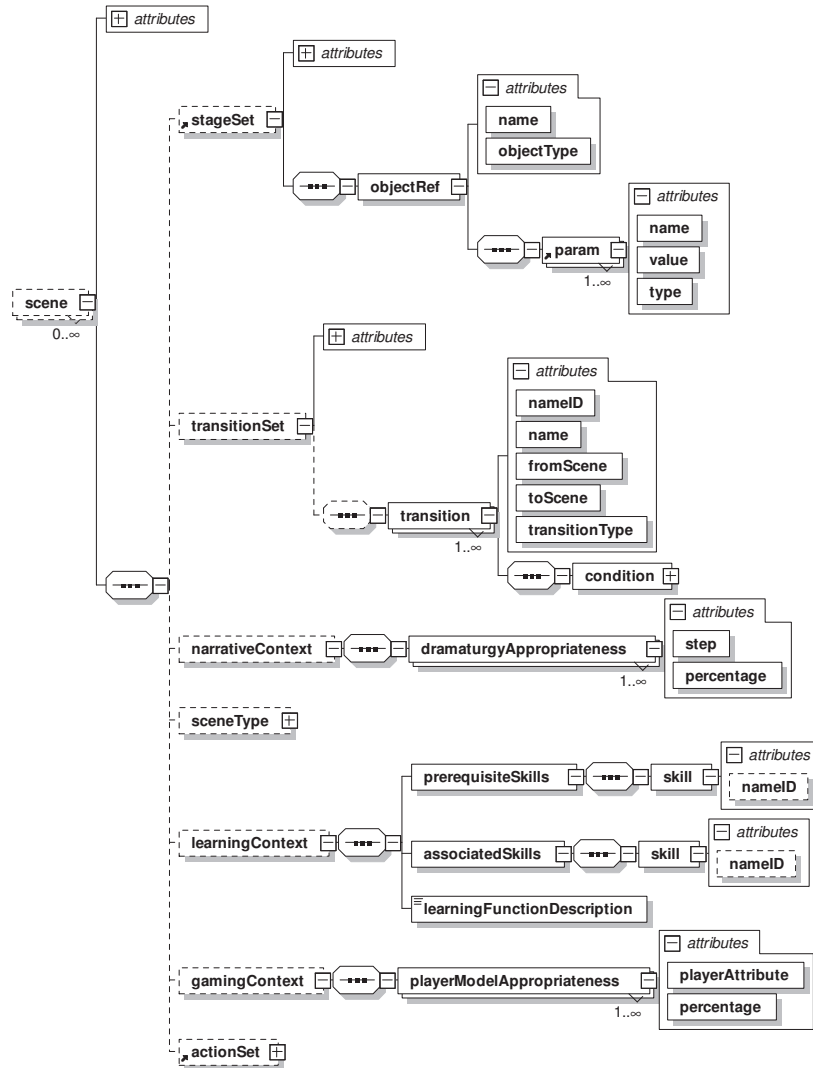


Figure 59: The schema definition of NGLOBs in ICML.

authors to check the effects of the adaptive algorithms at an early stage of the game's development.

Architecturally, StoryPlay is based on the Story Engine described above. Both parts of the user interface are connected to the generic interface of the Story Engine and receive updates from it. The visualization part of StoryPlay passively observes state changes in the Story Engine and visualizes them. The interactive part of the tool realizes the player application and therefore, apart from instantiating interaction templates, displaying content and executing actions sent by the Story Engine, also interprets player actions and converts this input into stimuli for the Story Engine when appropriate.

The individual visualization options of StoryPlay are described here.

### 7.6.1 Variables

The first visualization in StoryPlay displays all active variables in the game and their current values. Variables can be created locally inside a specific scene or on a global

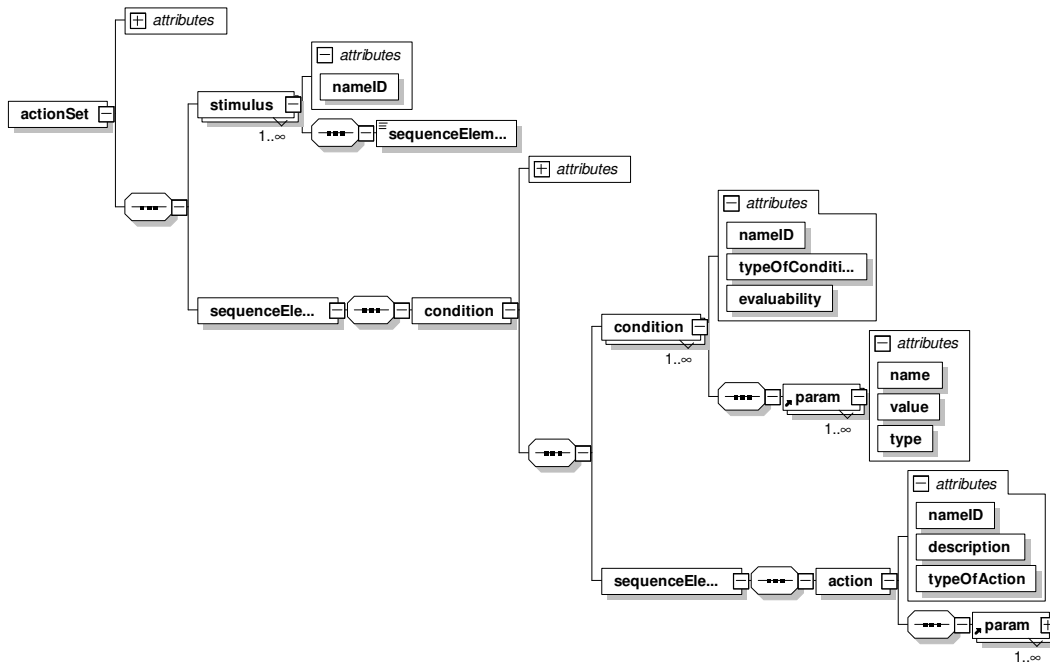


Figure 60: The schema definition of ActionSets in ICML.

level for the whole game. They can be used in various ways by authors, for example as flags (e.g. indicating that certain events have been triggered by the player) or as counters (e.g. counting the number of attempts a player had at task or how often a player has visited a certain scene).

### 7.6.2 History

The history visualization (figure 63) displays the history of past scenes and the current scene, along with the possible alternatives. Instead of using the possibly complex visualization of the Story Editor of the whole game, here a streamlined layout is chosen in which the scene is shown as a colored circle and transitions as simple lines, straight for normal transitions and dashed for free transitions. Whenever a scene is entered, it is highlighted in yellow, and all possible next scenes are added.

Whenever a scene has been annotated with the NGLOB information and a free transition leads to it, users can move the mouse pointer over the scene in question and get detailed information about the algorithmic parameters and how the rating of the scene based on the current models of the player would be. Therefore, authors can check the influence of their annotations in this visualization.

### 7.6.3 Narrative Context

In the narrative context visualization (see figure 64), the narrative aspect of the current NGLOB is shown. The steps of the used story model are arranged in their order on a horizontal line. Inside each step, the appropriateness values assigned by the author as a percentage are shown. Those values which exceed a threshold are marked in yellow, so one can immediately see which steps of the story model the scene is most appropriate for. Two shades of gray are further used in this visualization. The

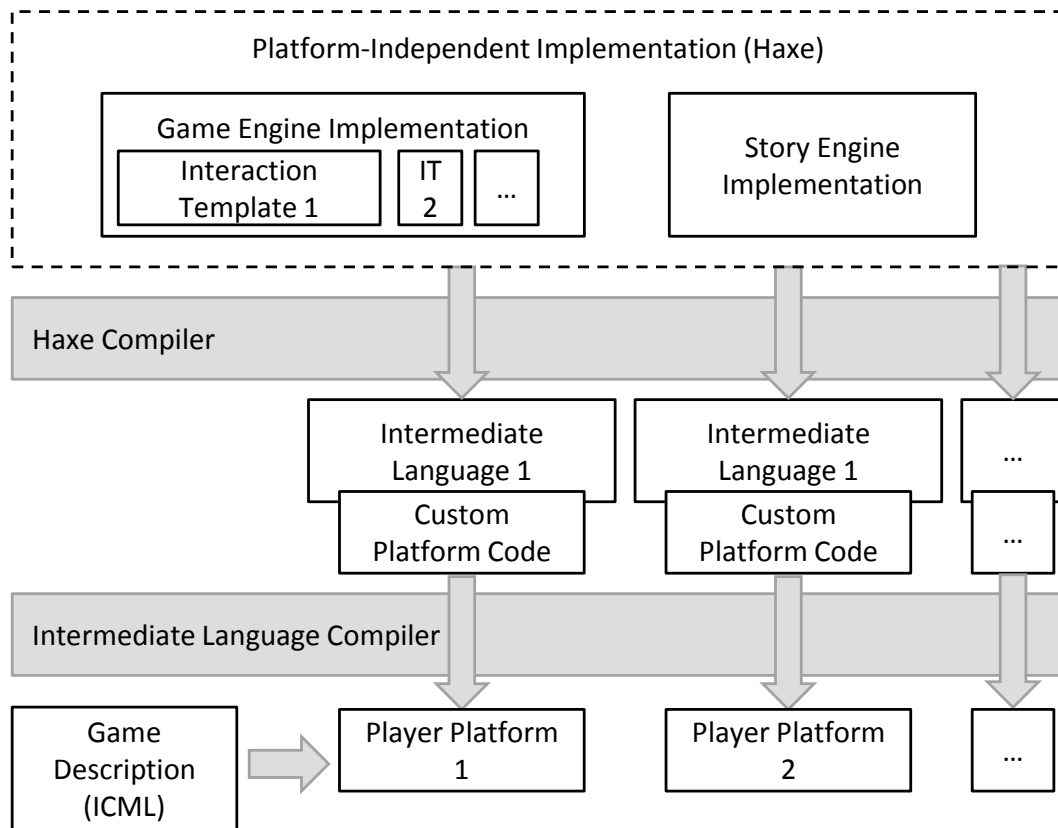


Figure 61: Cross-Platform development using the StoryTec approach.

default color is a dark gray, which is changed to a lighter grey when a scene appropriate for the respective story model step has been active in the game session so far. Therefore, by comparing the colors, testers can see whether steps have been skipped or are visited out of order.

#### 7.6.4 Player Model and Gaming Context

For visualizing the NGLOB aspects related to gaming, two similar visualization components are used. Both associate the categories of the player model (by default, the four categories of Bartle) with a percentage bar and a percentage value. The gaming context displays the author annotation for the scene. The player model visualization displays the current state of the internal player model, which is influenced by the updates as triggered by the author. It therefore allows testers to see the effects of different story paths on the player model.

#### 7.6.5 Skills and Knowledge Space

The learning aspect of NGLOBs is visualized by a section in StoryPlay dedicated to the observed levels of skills of the player. It is updated by the skill updates authors can trigger in response to stimuli from the game where appropriate. A textual list of skills shows the current values as numbers between 0 and 1. The list is divided into parts for the prerequisite and associated skills (set by the current scene annotation)

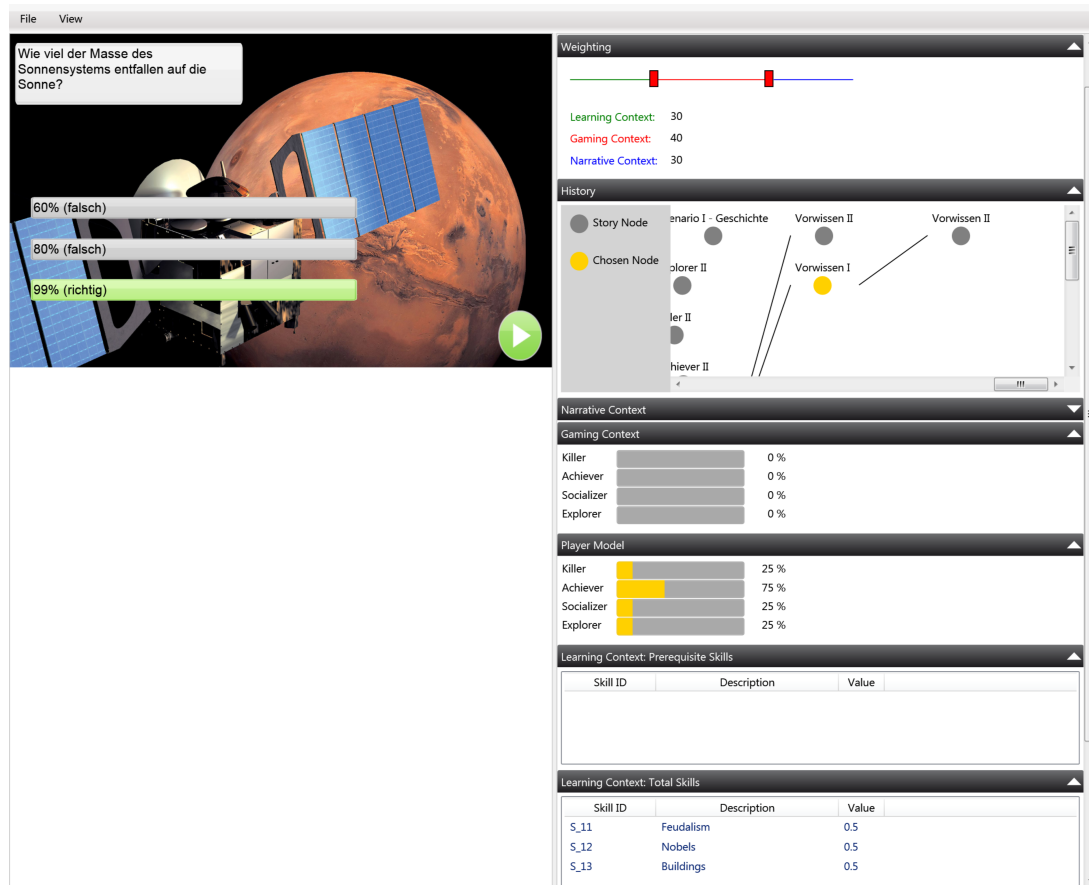


Figure 62: The rapid prototyping and analysis tool StoryPlay.

and the overall list of skills of the player (taken from the current state of the learner model).

As an alternative visual summary of the learning progress in the game, the graph created from the skills and the dependency relationships between them are visualized in the Knowledge Space visualization. Skills are shown as rectangles which display the name of the skill and a color field which changes color dependent on the level of the skill in the learner model. Colors are interpolated from red for 0 (no knowledge at all) to green for 1 (completely mastered). Thereby, the testers get a quick overview of how the gameplay can influence the learning progress and whether the skill structure and dependencies are correct. If this were not the case, one would see skills higher up in the graph colored green before the skills at the lower end had been learned.

## 7.7 CONCLUSION

In this chapter, the implementation of the concept outlined in chapter 3 and detailed in the previous three chapters has been shown.

The authoring tool StoryTec is the overall main result of this chapter. Building upon a framework providing a user interface and the implementation of the game model, a set of editors is included in the form of plugins. The four main components of StoryTec are the Story Editor, used for defining the structure of the game as a

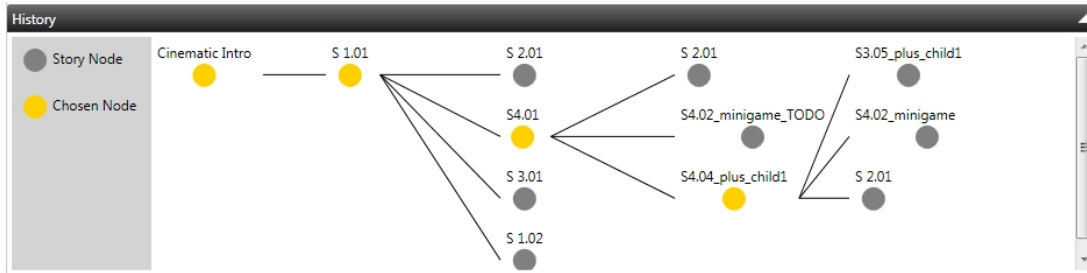


Figure 63: The history visualization of StoryPlay. Grey scenes indicate possible next scenes, yellow scenes highlight the scenes the player has been in.

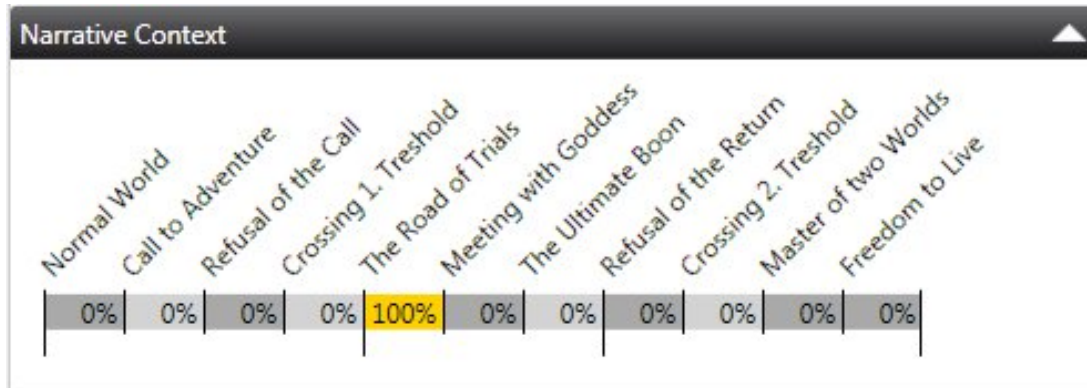


Figure 64: The visualization of narrative context in StoryPlay.

graph of connected scenes; the Stage Editor, for editing the details of each scene; the Objects Browser, which provides access to all object and template types; and the Property Editor, which allows editing of all parameters and metadata. Further components used dependently on the authoring context are the ActionSet Editor for defining the interactive elements of a game using a visual programming approach and the Condition Editor for executing parts of the game conditionally. Furthermore, a Skill Tree Editor allows authors to edit the knowledge structures used for adapting in educational games.

The authoring of adaptive games is specifically supported in StoryTec by including it in all appropriate components. Adaptive parts of a game are marked in the Story Editor, the Property Editor is used to annotate scenes with metadata necessary for adaptation and updates to the narrative, learner and player models can be made in the ActionSet Editor. While working with interaction templates, authors can be supported by providing software wizards which can be generated from the information provided along with interaction templates.

The design of StoryTec leads to an authoring process that can be applied in groups or for individual authors. In the case of collaborative work, a mapping of educational game development roles to components of the authoring tool is provided, showing that a complete workflow for the production of games can be realized with StoryTec.

A further result of this chapter is the architecture and implementation of a runtime platform for games that have been authored using StoryTec. The model description language ICML serves as the format for interchanging information between the authoring tool and the Story Engine. This re-usable component handles the execution of the game model, the management and updates of the NGLOB narrative, learning

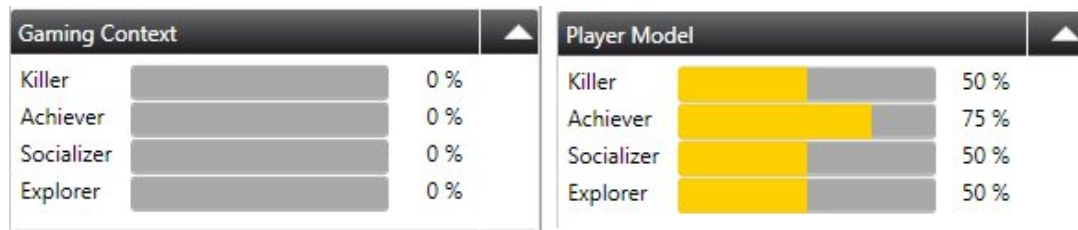


Figure 65: The visualization of gaming context (left) and the player model (right) in Story-Play.

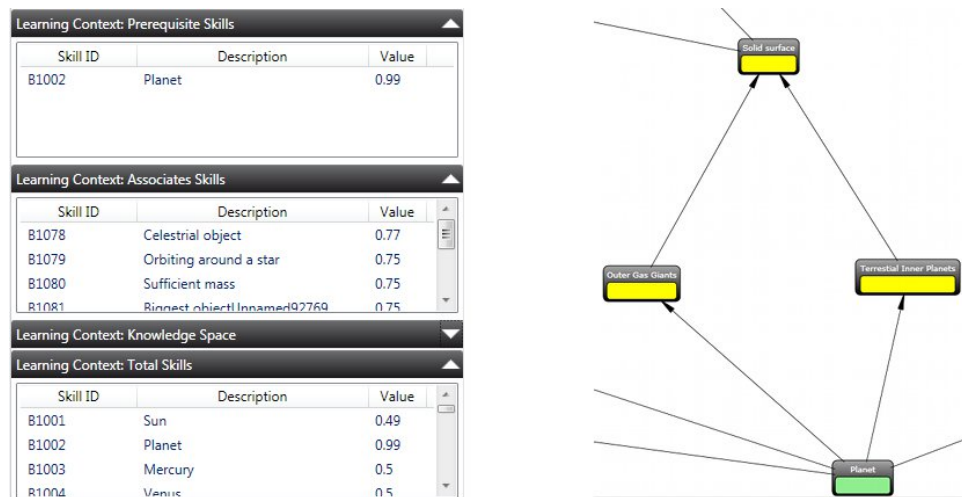


Figure 66: The visualizations for the skills of the player in textual form (left) and as a graph (right) in StoryPlay.

and player models. It is connected to a game engine, which realizes the interaction with the player and implements interaction templates. Information from the game engine is given to the Story Engine by the mechanism of stimuli, and the reverse communication towards the game engine is realized by sending the current scene, objects and actions.

This architecture is realized in two player applications, the cross-platform player StoryPublish and the rapid prototyping and scientific analysis tool StoryPlay. StoryPublish, based on the platform-independent programming language Haxe, allows a wide range of target platforms, among which versions for PC, Mac, Android, iOS and HTML 5 have been created and successfully used in several projects. StoryPlay is intended for rapid prototyping and analysis purposes and adds a set of visualization components which allow authors to examine the results of adaptivity during a playtesting session.

Implementation information about projects realized with StoryTec, either using the default runtime platform StoryPlay or integrating other game engines and platforms, is provided appendix A.

After demonstrating the technical realization of the concept of this thesis, the following chapter examines the evaluation of StoryTec and its applicability in the three application scenarios.





STORYTEC as the implementation of the authoring tool concepts has been evaluated in three extensive studies with users and several smaller studies. Before examining these studies in detail, we first discuss the various factors that can be observed in authoring tools and means for measuring them. Work in this field has not been carried out extensively and is in most cases restricted to user experience or usability studies.

- **Usability and user experience** are basic qualities of software products ensure that users are able to use software for their needs without long learning periods or without being restricted in their use of the software. Usability and a positive user experience is necessary for all three scenarios identified in chapter 3.
- **Efficiency** in the context of authoring tools relates to factors such as the time required to create a game using the authoring tool or the necessary effort to realize a specific design. Coupled with usability and user experience, high efficiency of the authoring tool is again a basic requirement for all use cases we identified.
- **Effectiveness** of an authoring tool can be seen as the degree to which an author is able to create a game matching the intended design and, correspondingly, how well the authoring tool is suited for the tasks it is worked on. By comparing a reference game to the result of an author in a study, we can infer the effectiveness of the authoring tool.

During the different phases of the development of StoryTec, the appropriate evaluation methods have been used. In the following, we present the results of the following studies:

- An initial usability evaluation carried out after one year of development, intended to elicit first feedback and directions for further development. This evaluation gave rise to a major re-design of the user interface.
- A follow-up usability study one year later, in which the changes from the previous study were appraised. The analysis of the results shows that the usability of StoryTec can be compared to common office software.
- A focus-group study with members of a commercial educational game developer and publisher. The purpose of this study was to assess whether the second scenario of game development teams (cf. 3.1 on page 39) is realistic. The results indicate that a fully developed version of StoryTec could be used in this scenario.
- A final comparison study in which StoryTec was compared with the similar authoring tool e-Adventure, the purpose of which was to assess the efficiency

and effectiveness of StoryTec and compare them to another tool. The results show that, after an initial learning period, authors can work faster and create more correct solutions with StoryTec than with e-Adventure.

The design of studies and questionnaires has been supported in the context of the cooperation of our research group with the department of Psychology.

### 8.1 INITIAL USABILITY STUDY

This usability study was carried out in order to test the first concepts and user interface specifications of StoryTec. Apart from assessing the general usability of StoryTec, a hypothesis of the study was that a reduction in complexity could be achieved by a reduced design (using fewer colors and more abstract icons).

For this study, we recruited  $N=29$  students, of which 3 were female, mainly from the Computer Science faculty.

The test was set up in a way that encouraged the participants to communicate and give feedback. It consisted of two phases. First, the participants tested the authoring tool based on a set of task for about 40 minutes. Secondly, a questionnaire was given out that the participants were asked to answer. Therefore, the test yielded both qualitative and well as quantitative feedback.

The study participants took part in the study in groups of three. Based on the “Think Aloud”-method [154], they were assigned the following roles:

- Reader: Participants with this role were in charge of reading out the tasks for the other two members
- Executor: This participant was in charge of the mouse and keyboard to actually manipulate objects in StoryTec, but was instructed to not act on his or her own
- Commander: Was in charge of giving instructions to the executor based on the task read aloud by the reader.

As can be seen, this separation of the members into roles enforced communication, since the person at the computer was not allowed to act on their own. Especially usability problems where expectations by the participants were not met or aspects that confused the participants became very clear during this phase. In order to garner the feedback, the test was recorded on video (with the permission of the participants) and by writing a protocol.

The tasks that the authors were given resulted in a small game that could be tested afterwards by the participants. The tasks included basic objectives such as creating a scene or a transition and adding content to a scene.

The questionnaire that was given to the participants after they had completed the practical part of the study was based on the ISO Standard EN ISO 9241-110 [63] and contained questions related to the seven fields of software ergonomics identified in the standard:

1. Suitability for the task
2. Self-descriptiveness
3. Controllability

4. Conformity with user expectations
5. Error tolerance
6. Suitability for individualization
7. Suitability for learning

The quantitative test consisted of a questionnaire with 35 items which were rated with a six point scale ranging from “very bad” to “very good”. For each of the seven areas of usability defined in ISO 9241-110, five items were included. See appendix B for the questionnaires for the studies described in this chapter.

In sum, the participants rated the interface, the offered functions and the concept of StoryTec positively. StoryTec was received well by most of the participants, with some of them describing it as an innovative, user-friendly authoring tool. No group lost motivation during testing and all tried to fulfill the given tasks to the end.

Problems became apparent in some areas of usability and technical bugs, which were due to the early state of the authoring tool. The most problematic component of the software at the time was identified when participants were unable to understand the function of the Stage Editor.

The test further indicated that the hypothesis of reduced complexity due to a reduced visual design was not correct, as the analysis showed that this reduced design lead to problems with understanding the user interface and the offered functions.

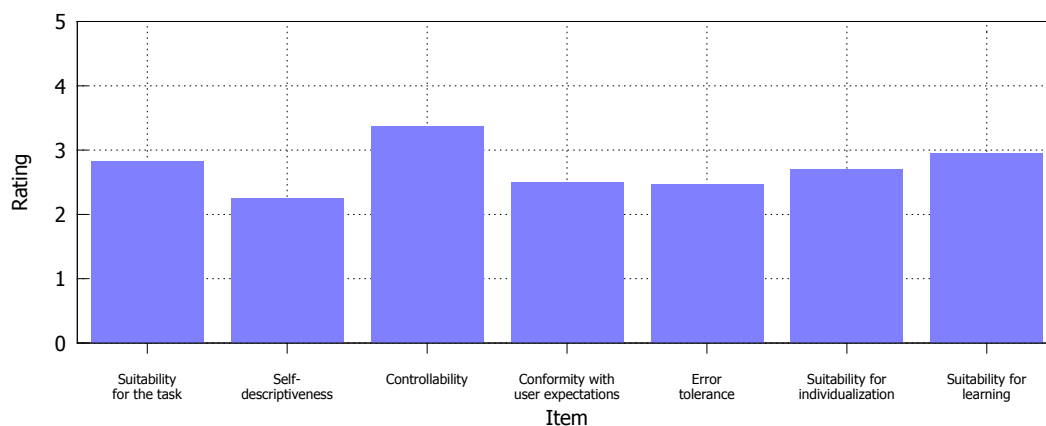


Figure 67: The average results of the questionnaire aggregated by the seven fields of ISO 9241-10.

Figure 67 is a diagram in which each of the five questionnaire items per field of usability defined in ISO 9241-110 are aggregated into one value. We see that no conclusive results can be drawn from this overview since all values are in the same range and analysis shows a high standard deviation.

As an overall result, the main outcome of this first usability evaluation is in the qualitative feedback, where the “Think Aloud” method yielded many insights into the usability of the authoring tool and lead to a comprehensive update in the user interface design and new paradigms in which the authoring tool was used. The initial version of the Story Editor, re-used as a component from the INSCAPE authoring tool, was identified as one source of usability problems as many study participants were observed to have problems with the foundational task of drawing transitions

between scenes. Therefore, a new version of the Story Editor was created and evaluated in the follow-up usability study. A related problem that became apparent were “complex scenes”, referring to scenes that are used to hierarchically organize other scenes. The participants did not seem to understand this concept and stated their confusion about this concept.

## 8.2 FOLLOW-UP USABILITY STUDY

A second usability study was carried out a year after the first usability study described above, in order to test the updated version of StoryTec. Among other changes, a completely new Story Editor was created to address the shortcomings identified in the previous usability study. Furthermore, the concept of “complex scenes” was removed, replacing it with only one type of scene as described in chapter 4.

In order to provide comparability of the results, the experiment design was adapted to yield results comparable to those by Prümper [116]. Prümper carried out a large-scale study, evaluating the usability of 41 software tools with 1265 participants.

The participants of this study were recruited from a lecture on Serious Games.  $N=26$  participants were included, one of which was female. The mean age of the participants was 25.2 years ( $SD=3.71$  years).

The study setup was similar to the previous usability study, with participants first working on a game and then filling out a questionnaire. The task the participants were given was drawn from the Geograficus educational adventure game modeled in StoryTec (cf. A.3). While performing the test, previously deleted parts of the game had to be re-created by the participants. For the practical part, the participants were again sorted into the roles “reader”, “executor” and “commander” to elicit feedback.

The questionnaire included questions on the previous experiences of the participants with authoring tools and items to assess the usability areas of ISO 9241-110.

13 participants claimed previous experience with authoring tools, including “Unity 3D”, “Blender”, “3ds Max”, “Audacity”, “Novelty”, “RPG Maker” and “Photoshop”. They were also asked to rate their knowledge of these tools. The perception of this knowledge varied strongly: on a scale from 1 to 7 (1 indicating no knowledge, 7 indicating mastery), a value of  $m=3.69$  ( $SD=1.70$ ) was reached.

Figure 68 provides the results of the questionnaire, grouped again by the seven usability areas defined in the standard.

The questionnaire also included a question for the perceived knowledge of StoryTec after the test. The average result to this item was  $m=2.64$  ( $SD=1.44$ ). This value is lower than the knowledge the participants claimed for the other authoring tools. However, we have to keep in mind that this perception was the result of only one hour of working with StoryTec.

When comparing the results of the questionnaire with those of Prümper, we see that the ratings achieved in this study are similar to those of Prümper, apart from the two areas self-description and error tolerance. This indicates that the usability of StoryTec is comparable to other software products.

The large amount of problems with the Story Editor encountered in the previous usability study did not manifest in this study, which indicates that the new version of the Story Editor is more usable and intuitive.

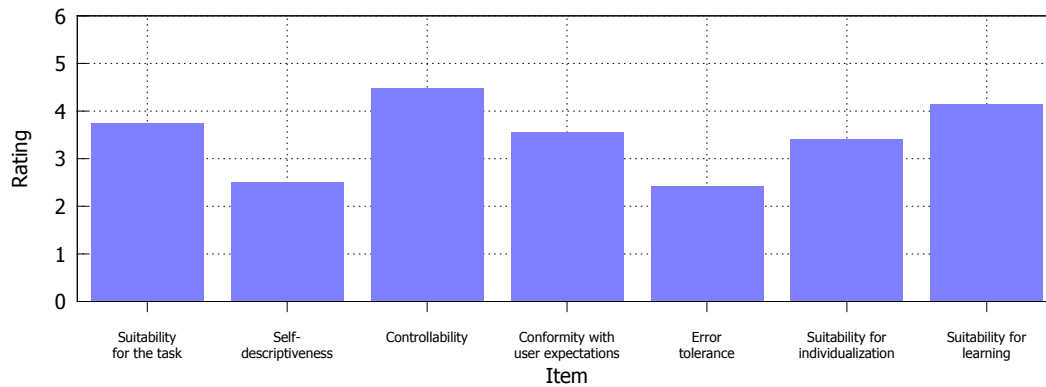


Figure 68: The average results of the questionnaire aggregated by the seven fields of ISO 9241-110.

The lower score in error tolerance can be explained with the occurrence of technical problems in the prototypical version of StoryTec that was used in the study. The aspect of self-description has been addressed in later versions of StoryTec by adding more descriptive texts as well as a tutorial accessible from the main user interface.

### 8.3 FOCUS GROUP STUDY WITH GAME DEVELOPERS

After the previous two usability studies had demonstrated that novice authors could work with StoryTec sufficiently and several general usability issues that had become visible were addressed, a focus group study was planned and executed in order to examine the applicability of StoryTec in actual game development. The hypothesis for this study was that game developers could use StoryTec in the development of educational games, as captured in the second scenario in section 3.1 on page 39.

For this purpose, the focus group study was carried out in cooperation with Braingame, a commercial German developer and publisher of educational games, including educational adventure games. Three members of the company involved with the game development tasks participated in this study. The participants were aged 31, 37 and 46, one participant was female, two male.

This study commenced with a set of tasks similar to the previous two usability studies, presenting an incomplete part of an educational game that the participants were asked to work on. Since the test was carried out with one participant at a time, the tasks were adapted to be carried out by only one person. However, to encourage feedback similar to the group-based “Think aloud” method used previously, two investigators were present and elicited feedback by asking what the participants were thinking or how they were trying to achieve their goals in the software.

After completing the tasks, the investigators lead the participants through a guided interview. The interview included a set of 15 questions, which were based on the criteria of ISO 9241-110. Apart from assessing the usability of StoryTec, the questions were also intended to give information about the applicability of StoryTec in the participants’ respective fields (including game design, art and programming). Furthermore, the participants were asked to assess the applicability of StoryTec for teachers

creating educational games for classroom use on the one hand and game designers on the other had.

The qualitative results of this interview included the experts' general interest in the StoryTec approach and their estimation that a full version of StoryTec might be used in game development. In the prototypical state that they were exposed to, they claimed to regard it fit for storyboarding and prototyping tasks.

All interviewees stated that they could imagine StoryTec being used by a teacher more than a game designer. This choice was motivated by one participant by the factors that the visual programming approach in StoryTec was simple and that the genre of point & click adventure games was a good choice. A German translation was called for in the scenario of teachers using StoryTec on their own (scenario 1, see chapter 3.1), since the English interface of the tool would interfere with their understanding.

The participants perceived the functions of StoryTec in the version of the test as too limiting for a professional game designer. The exception to this was the use as a tool for creating storyboards, i.e. non-interactive or minimally interactive overviews of the game which establish its art style and narrative. This use would require more possibilities for taking notes and text capabilities as one participant pointed out. The participants also stated that the presented version might be well suited for the needs of hobbyist game designers.

When asked to estimate how much time the participants would need to understand all functions of StoryTec and use it productively, two participants stated "two hours" while one claimed "several days spent building more complex games". This indicates that the short exposure to StoryTec in the course of the study (less than one hour) allowed them to learn most of the functions and understand them. Concerning specific functions, all participants gave feedback that they were not confident that they used the Story and ActionSet Editor in the intended way. This feedback has given rise to later changes in StoryTec. For example, we added context-sensitive commands to these editors in the form of context menus, since the participants were expecting this functionality.

#### 8.4 COMPARATIVE STUDY WITH E-ADVENTURE

While the previous usability studies could show that the usability of StoryTec is comparable to common office software, they have examined the tool in isolation and were not intended to elicit information about the efficiency and effectiveness of StoryTec. In order to evaluate these parameters, a study comparing StoryTec with the e-Adventure authoring tool (see section 2.3.3 on page 21) was carried out.

The hypothesis underlying this study was that untrained authors would be able to work with StoryTec with higher efficiency and effectiveness than with a-Adventure, due to different approaches, especially concerning the game model, the interaction design with the authoring tool and due to the author support of StoryTec, including interaction templates.

The experiment set-up consisted of a task that was phrased to be equally accomplishable in both tools which consisted of three tasks moving from simple to more complex interaction with the respective authoring tool. During each evaluation session, a group of up to 8 participants (each with an individual PC) was instructed

to work for 25 minutes on the tasks in the first authoring tool and then for 25 minutes in the other authoring tool. The order in which the tools were evaluated was randomized per group of participants. After concluding this step, the participants were asked to fill out a questionnaire with individual sections for each authoring tool, again based on the areas of the ISO 9241-110 standard. Additionally, some background information, an assessment of the perceived level of mastery of the respective authoring tool and a comparative question between the authoring tools was asked for. During the two authoring sessions, the participants were asked not to communicate with each other and no hints concerning either conceptual or technical problems were given by the investigator for either authoring tool.

The three tasks covered the following skills in the use of either authoring tool:

1. Parameter change
  - a) Identify a scene by its name
  - b) Change the background image of a scene
2. Basic navigation
  - a) Create a new scene
  - b) Add a background image to the scene
  - c) Create an interactive area in the scene
  - d) Link the scene to other scenes correctly
  - e) Make the scene the default entry point of the game
3. Complex interaction
  - a) Create an interactive area that can be used in a puzzle where a set of images is shown and the correct combination has to be entered

While the first and second task were similar in the nature of their solution in StoryTec as well as e-Adventure, the third task required different approaches in the two authoring tools. Therefore, task 3 is not further broken down into individual skills since the solutions in the two authoring tools diverge and do not require the same or comparable skills. The task description can be found in [appendix B](#).

The first task could be solved in very similar fashion in both tools. In StoryTec, it required navigating in the StoryEditor, selecting the “Background Image” object of the scene and changing its “File” property. In e-Adventure, the participants could find the scene in the list of scenes or in e-Adventure’s Story Editor.

The second task started to diverge between the tools. While StoryTec offers the addition of a new scene in the menu of the StoryEditor as well as in the context menu accessed by right clicking, e-Adventure only features a “plus”-button in the list of scenes. Especially since e-Adventure’s Story Editor has only limited editing capabilities, it does not allow for adding new scenes directly in the editor. Task 2b diverged insofar that authors had to add the background image object in StoryTec first and then set it to point to an image, while the participants using e-Adventure could re-use their workflow from task 1. Task 2c diverged in that StoryTec treats hotspots for exiting a scene the same as all other objects and allows adding them in the same way through the objects browser, while e-Adventure has several tabs of which only one is dedicated to exits. Task 2d required StoryTec authors to draw



transitions from the new scene to the connected scenes first before choosing them as exits. On the other hand, e-Adventure exits could be set up immediately, but the target scenes had to be chosen from a small, non-resizeable drop-down menu with all scenes. Task 2e can be solved in StoryTec in the StoryEditor (either in the context menu or the editor's menu) while the setting can be found in e-Adventure in the "chapter settings".

The third task diverged the most, since the solution in StoryTec is based on an interaction template while the only possible solution in e-Adventure lies in utilizing e-Adventure's visual programming language. The scene in question was set up to contain an interaction template providing an "ImagesHotspot", an object that can be set up with a set of images that are cycled and that trigger a stimulus when all images in the scene have been correctly set. Therefore, the solution entailed only adding a new ImagesHotspot element and loading the right images. On the other hand, in e-Adventure, this is realized by a number of "set items" (images that can be placed in the scenery that are not clickable) and a number of "active areas" which, upon clicking them, cycle a variable that in turn indicates which of the set items is to be shown. Most of these steps had to be repeated several times by the participants in order to complete the task successfully.

Apart from the quantitative data captured in the questionnaire, the experiment was also set up to elicit performance data from the participants. They were asked to save the state of their game whenever they felt that they either had completed the task fully or they were unable to continue without help. To prevent users from overwriting previous versions and thereby invalidating time measurements and the resulting files, we equipped both StoryTec and e-Adventure with buttons that allowed a write-once save function for the three tasks to a folder hidden from the study participants. Furthermore, we measured the time differences on the used computers to correct for differences. A set of criteria was prepared that was later used to determine the correctness and completeness of the tasks in the project files.

A set of  $N=47$  test subjects was recruited for this comparison study from a university course on Serious Games. Of the 47 participants, 8 were female and 39 male, aged between 21 and 32 years ( $m=24.79$ ,  $SD=2.62$ ). As described in the study setup, the order in which the two authoring tools were used was randomized, with  $n(1)=25$  participants starting with StoryTec and  $n(2)=22$  starting with e-Adventure.

Results of analyzing the questions in the questionnaire indicate that the participants preferred StoryTec ( $m=4.58$ ,  $SD=1.17$ ) to e-Adventure ( $m=4.21$ ,  $SD=0.78$ ) ( $p=.084$ ). We found an interaction between gender (male, female) and authoring tool rating (StoryTec, e-Adventure). Male participants rated StoryTec higher ( $p=.023$ ) than female participants, who tended not to see significant differences between the tools ( $p>.20$ ). Statistical analysis shows that this interaction borders significant ( $p=.072$ ).

The project files created during the test were saved and later analyzed using an automated workflow for retrieving the objective time measurements and a semi-automated workflow for rating the criteria for completeness and correctness of the three tasks. In this workflow, the simple tasks were automatically checked and heuristics indicated which solutions to the more complex tasks could be excluded from manual inspection due to apparent errors. The remaining cases were rated manually by an investigator. Due to technical difficulties during the test, four data points for the third task from StoryTec are not available and are therefore excluded from the following calculations.

The average results of the time measurements are shown in figure 69. When comparing the measurements, we see that participants spent 58% more time on the first task and 18% more time on the second task in StoryTec compared to e-Adventure. However, this is reversed in the case of the third task, for which participants using StoryTec only required 55% of the time spent in e-Adventure. As an overall result, due to the shorter time working on task three, StoryTec users finished the overall tasks earlier than e-Adventure users and finished before the end of the 25 minute duration of the test. (StoryTec:  $m=1409$  seconds,  $SD=296$  seconds, e-Adventure:  $m=1535$  seconds,  $SD=181$  seconds).

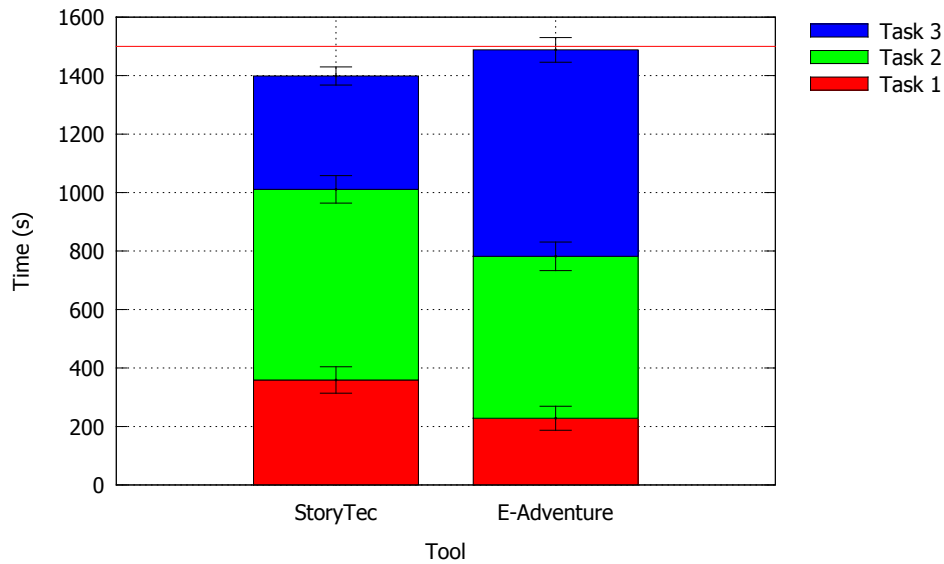


Figure 69: Comparison of average times spent on the three tasks in the two authoring tools. The red line indicates the end of the test after 25 minutes. Error bars indicate one standard error.

The set of criteria for rating the completeness and correctness of the participants' was applied to the available data files. These criteria allowed minor differences to the task instructions in both authoring tools, mainly relating to typing errors in the naming of objects or scenes. Figure 69 shows a comparison of the ratio of participants that were able to complete each of the tasks in the two authoring tools completely and correctly. As can be seen, task one was completed very well in both authoring tools. On task two, the results started to diverge, with 52% of participants solving the task correctly and completely in StoryTec and only 30% in e-Adventure. 57% of participants were able to solve task three using StoryTec, compared to 24% using e-Adventure. Further analysis of the project files shows that 7 users (17%) in StoryTec committed only minor errors in StoryTec while working on task three.

As mentioned above, a semi-automatic process was used to rule out project files for task three that could not be correct by definition. Conversely, the same logic can be applied to find out which participants showed that they had the correct idea how to carry out task three. From this analysis, we result at 68% of participants who showed the correct idea in StoryTec, of which only 25% failed while carrying out their plan. 55% of e-Adventure users showed they had understood the basic method of carrying out task three, but 44% of these users failed along the way of implementing their solution.

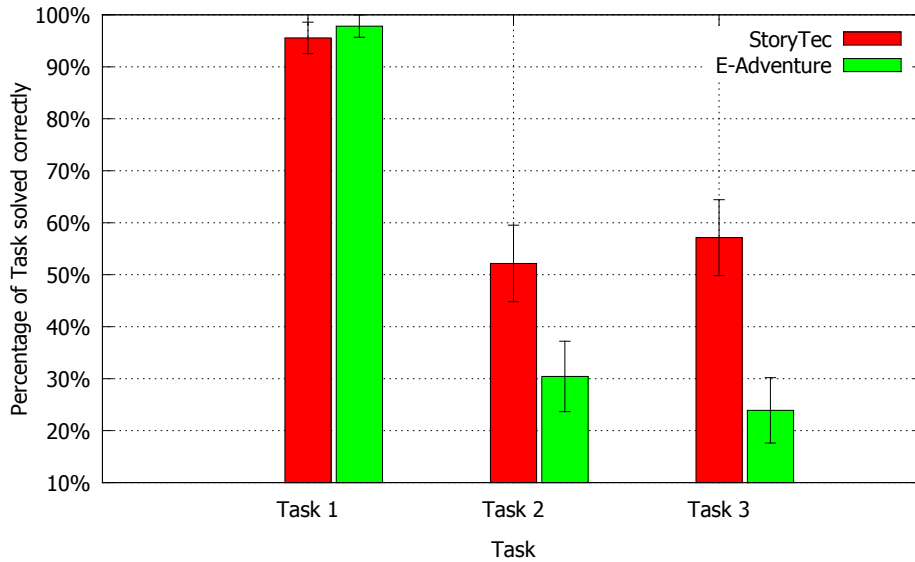


Figure 70: Comparison of the average percentage to which tasks were completed and correctly finished in the two authoring tools. Error bars indicate one standard error.

For a discussion of these results, we first turn towards comparing the times required for the three tasks in e-Adventure and StoryTec. We can observe that users required more initial training time in StoryTec than e-Adventure. One possible explanation for this is the difference in user interfaces. StoryTec uses a customized style and follows the user interaction principles of applications from creative or content creation applications while e-Adventure uses an interface more in line with regular desktop or office applications. Furthermore, as appendix A demonstrates, StoryTec is also applicable to other genres of Serious Games apart from educational games and therefore offers more features than e-Adventure, which is limited to the genre of educational adventure games.

During the comparative study, we intended users to re-create parts of the game based on examples they found in the remaining parts of the game. The quickest way to carry this out lies in using the copy and paste functionality, which we observed several users to utilize (however, no formal measurement of the amount of users preferring copy and paste to the alternative way of building the game elements “from scratch” was included in the study design). We can conclude from this observation that this feature is helpful for authors and should be supported. This also underlines the concept of structural templates, which are similar to copy and paste since they also rely on replicating previously configured parts of a game.

Analysis of the sub-tasks of task two shows that the participants had minor problems creating a scene and adding transitions in both tools. However, a large part of e-Adventure-users failed in changing the start scene of the game. This setting is offered in a context menu for each scene in StoryTec as well as in the form of a command of the Story Editor. On the other hand, this setting is a property of a “chapter” in e-Adventure, where many participants did not expect this setting and consequently did not find it. This corroborates the central role of the Story Editor in StoryTec, in that it offers all functions related to the game structure in one central location.

Concerning the third task, we see that the solution to this problem differs between the authoring tools. e-Adventure users had to re-create the game mechanics in the third task by using the programming language of e-Adventure and by making extensive use of its “conditions” and “effects”. A similar, programming-oriented solution was possible in StoryTec using the ActionSet Editor. However, StoryTec also provided an interaction template for the gameplay found in the third task, which all participants who worked on this task chose to use.

Many of the participants failed in correctly using the e-Adventure programming facilities, even though the complete solution could be understood by examining the existing parts of the game which had the game logic set up correctly. However, even though some participants as described above started in a correct way, many of them failed in emulating the existing solution correctly. We conclude that the concept of dedicated interaction templates assisted authors in quickly and correctly understanding the authoring process of the game and to carry it out on their own. We suggest that model checking as described in chapter 6.4 could increase this effect even more especially for games in which the visual programming language is used more extensively, since it is designed to make authors aware of errors such as those committed by the participants in this study.

## 8.5 CONCLUSION

We identified the effects of usability and user experience, efficiency, effectiveness and quality of the authored games as relevant to the applicability of StoryTec for the three scenarios addressed in this thesis.

In two usability studies, this basic requirement for all uses of the authoring tool has been evaluated. An initial study evaluating the usability of StoryTec based on the ISO 9241-110 standard was carried out, resulting in qualitative feedback that was used in the further development of StoryTec. Relevant usability improvements were the exclusion of “complex scenes” which were identified as a superfluous and confusing element of the game model, a new version of the Story Editor and an update to the visual design of StoryTec since an initial, reduced look did not help authors to understand the tool.

A follow-up usability study was carried out in order to examine the effects of changes introduced as a response to the first usability study. Furthermore, for reasons of comparability with a large-scale study of office software, the experiment design was adapted to allow comparison. The usability study yielded the result that the usability of the StoryTec prototype is average compared to other office software.

In order to ascertain whether StoryTec could be used in the context of game development teams, a focus group study with a German educational game development studio was carried out, evaluating the usability and the applicability in this scenario. The results indicate that the prototype version of StoryTec used in this study could be used in a limited setting, with the areas of storyboarding for educational games or amateur game development being mentioned as a use case. Furthermore, the study participants declared that a fully developed version of StoryTec might be used in commercial game development.

A final comparative study was designed and carried out in order to allow a comparison of usability, efficiency and effectiveness of StoryTec and the related educa-

tional adventure game authoring tool e-Adventure. This study diverged from previous studies that quantitative data was not only collected by means of a questionnaire but also as performance data captured by the software. The results of this study showed that StoryTec requires a larger amount of initial training time, however, authors using StoryTec are able to carry out complex authoring tasks in less time and with a higher degree of correctness. Especially the concept of interaction templates can be seen as an advantage of StoryTec compared to other authoring tools.

Apart from the studies described here, the applicability of StoryTec in game development has been demonstrated in the context of several research projects where StoryTec has been used as a foundational system. Several educational games have been authored with StoryTec, including the demonstrator of the EU-funded project “80Days”, the educational game for mathematics “Der Wechsel” and a re-authored version of the commercial educational adventure game “Geograficus”. Further projects have shown that StoryTec can also be successfully used in other Serious Game areas, for example with the health game “Motiation 60+” for elderly people, where StoryTec is used to configure and customize both quizzes for cognitive training as well as physical exercises. Another example is the advertisement game “Der Chaos-Fluch” used by the city of Darmstadt for marketing purposes. All projects and games mentioned here are described in detail in appendix [A](#).

## CONCLUSION AND OUTLOOK

---

THIS thesis examined the development of adaptive educational games, which promise to be effective in learning settings due to their high appeal, inherent motivation and adaptation to players' needs. Based on a comprehensive analysis of the related work as well as studies with potential users, the need for an authoring tool for such games along with a set of requirements have been identified. The development processes that are currently in place for digital games in general and especially for educational games were characterized as complex, collaborative work that include both experts from multiple domains. Educational games were identified as especially complex to develop, since they often have higher requirements in some areas such as authenticity or content quality, but often have to cope with smaller budgets and team sizes than entertainment games. Adaptivity adds further complications by requiring additional, adaptable content and by increasing the difficulty of testing a game.

We identified three scenarios in which an authoring tool for educational games could be introduced. In the first scenario, a single author creates a game without assistance by others. In this scenario, the main requirement for the authoring tool is a low barrier to learn and use it. This should be achieved by ensuring the usability of the tool and by lowering the necessary skills to use it, especially by providing an alternative to programming languages. The needs of users in this scenario were identified by carrying out a user study with teachers. The second scenario features a group working collaboratively on an educational game, a situation commonly found in game development studios. Since users have diverse specializations and qualifications, the provision of a collaborative workflow is key in this scenario. The requirements of users in this scenario were assembled from interactions with an educational game development studio. The third scenario is defined by separating between an author creating a game and an author customizing the game at a later stage. This scenario is exemplified by situations where a game is produced and later personalized for a specific player, for example with knowledge about this player's specific preferences or needs. In this scenario, the main requirement is the specific support for the customizing authors, who should not be burdened with details which are outside of their scope. The requirements for this scenario have been drawn from conversation with users who might use this functionality, such as teachers or therapists.

Based on the identified requirements and the related work, a concept for a novel authoring tool has been elaborated. We arrived at the concept by means of user-centered design processes and the context of applied research in the field of educational games. The major components of this concept are a model for educational game structures and content, technologies for allowing the authoring of adaptive games and several means of author support. As a basis for the whole authoring tool concept, a model that can capture all relevant aspects of an educational game has been presented. This model allows an authoring tool to present this model to an author in an intuitive way, replacing structure definitions and programming language

code by visual representations. The second layer of the concept accounts for the authoring of adaptive games that are able to configure themselves during play to be optimal for a certain player. To this end, we analyzed the existing models for the key areas storytelling, learning and gaming and combined them into the concept of Narrative Game-Based Learning Objects which were presented as an extension of the basic game model. We furthermore used the model to describe four means of supporting authors of an authoring tool for educational games. Structural and interaction templates provide re-usable structures that can be used to quickly author games based on configurable game elements. Support for collaborative authoring is realized by the definition of specialized views for different users groups by specifying individual access and visibility levels to game objects. Iterative development and rapid prototyping are included in the concept by specifying a testing tool suited for these tasks. By applying model checking to game models, authors can be warned for several categories of semantic errors such as insufficient metadata for adaptation.

The concept for an adaptive, educational game authoring tool was implemented as StoryTec, complemented by the multi-platform player StoryPublish and the rapid prototyping and scientific analysis tool StoryPlay. We demonstrated how the aspects of the concept, including the adaptive game features and interaction templates, have been integrated into this authoring tool. Furthermore, the identified roles in educational game development have been mapped into the components of the authoring framework, demonstrating a workflow for game creation using StoryTec.

We demonstrated the appropriateness of the concept and the usability of the implementation in a series of evaluation studies. Two initial usability studies during the first development phases of the authoring tool yielded useful insight into the way authors work with the authoring tool and have given rise to design and style changes. Furthermore, they showed the implementation to have usability comparable to other common software. The applicability of StoryTec in actual use has been further demonstrated in several research projects, building a foundation for current and future approaches in the field of Serious Games. An active open community of users testing and applying StoryTec has been established, consisting of over 120 users.

A focus group study with a German educational game developer showed the possible applications of the authoring tool in professional game development, with the outlook of a final and complete version being usable in actual commercial game development.

Finally, a comparison study, comparing the authoring tool with the related authoring tool e-Adventure, allowed us to draw several conclusions. While StoryTec has a higher training time for basic tasks than e-Adventure, we find that novice authors can solve these basic tasks in a similar quality. For more complex tasks that touch on game aspects that require more algorithmic thinking to realize, we have observed the positive effect of the interaction template concept on the quality and speed of the solutions in StoryTec.

The work presented here can be extended and applied in the following areas:

**PROCEDURAL CONTENT GENERATION** The choice of a game model underlying the authoring tool and the use of a visual programming approach has allowed authors without programming skills to work on games. However, as has been seen in many examples of the use of StoryTec, the creation of game content is now

the bottleneck for game creation. We therefore propose that future research could address the integration of techniques for procedural content generation in the authoring workflow. These techniques allow non-artists to create content from a set of rules that have been previously encoded.

**SEMANTIC CONTENT RETRIEVAL** By applying semantic content retrieval to authoring tools would allowing authors to specify the content they want and then draw it from existing databases. Both in procedural content generation and semantic retrieval, the work on game modeling has to be intensified in order to cover specific content, in addition to the already covered structure, to allow the description and subsequent generation or retrieval of content. More in-depth application of these techniques could also be applied directly to the game itself, leading to semi-automatic game design of Serious Games. Especially the authoring process of adaptive games could benefit largely from this, since authors could provide one version of the game as a template and let variations for different player types or learning paths be created semi-automatically.

**NOVEL AUTHORING PARADIGMS** Current games have given rise to novel concepts such as in-game editing, where players manipulate aspects of the game directly in the game world, user-generated content, where players exchange game content in communities associated with one game, and gamification concepts, where normal tasks such as the creation of games are games themselves. The game “Minecraft”(presented in section 2.1 provides an example of the combination of procedural content generation and in-game editing, and has allowed hundreds of players that are not trained artists to create intricate 3D worlds and objects from very basic primitives. Departing from current paradigms such as timeline-based authoring towards such novel paradigms could create a direct interface to the final game, allowing authors to get immediate feedback on their work.

**MULTI-PLAYER GAME AUTHORING** The work presented here has concentrated on laying a foundation for authoring of single-player Serious Games. However, the effects of Serious Games could be enhanced by providing games for several players. While an extension for multiplayer Serious Games is therefore desirable, we also see that such games have ramifications on all aspects of an authoring tool. The design of the game has to be adapted to cater to several players, creating situations that promote cooperation. The game model has to account for events happening in parallel, conditions and actions are required to have potentially different effects on each player, and adaptivity has to be applied to a group as well as to an individual player. An initial version of StoryTec with rudimentary multiplayer authoring support has been created [121] and research in this area will be intensified in the future.





## BIBLIOGRAPHY

---

- [1] Advanced Distributed Learning. Scorm 2004 4th edition, 2009.
- [2] Dietrich Albert and Josef Lukas. *Knowledge spaces: theories, empirical research, and applications*. Psychology Press, 1999. ISBN 0805827994.
- [3] Michael W. Allen. *Michael Allen's Guide to E-Learning*. Wiley, first edition, 2002. ISBN 0471203025.
- [4] Alan Amory. Building an Educational Adventure Game: Theory, Design and Lessons. *Journal Of Interactive Learning Research*, 12(2):249–263, 2001.
- [5] Ruth Aylett. Narrative in Virtual Environments - Towards Emergent Narrative. In M. Mateas and P. Sengers, editors, *Proceedings 1999 AAAI Fall Symposium on Narrative Intelligence*, Working Note of the Narrative Intelligence Symposium, pages 83–86. The AAAI Press, 1999.
- [6] Christine Bailey and Michael Katchabaw. An Experimental Testbed to Enable Auto-Dynamic Difficulty in Modern Video Games. In *2005 GameOn North America Conference*, pages 18–22, 2005.
- [7] Tom Baranowski, Richard Buday, Debbe I. Thompson, and Janice Baranowski. Playing for Real: Video Games and Stories for Health-Related Behavior Change. *American Journal of Preventive Medicine*, 34(1):74–82, 2008. ISSN 0749-3797.
- [8] Richard Bartle. Hearts, clubs, diamonds, spades: Players who suit MUDs. *Journal of MUD research*, 1(1), 1996.
- [9] Chris Bateman and Richard Boon. *21st Century Game Design*. Charles River Media, 2005. ISBN 1584504293.
- [10] Francesco Bellotti, Riccardo Berta, Alessandro De Gloria, and Ludovica Primavera. Adaptive Experience Engine for Serious Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 1(4):264–280, 2009. ISSN 1943068X.
- [11] Erik Bethke. *Game Development and Production*. Wordware Publishing Inc., 2003.
- [12] Ahmed BinSubaih and Steve Maddock. Game portability using a service-oriented approach. *International Journal of Computer Games Technology*, 2008, 2008.
- [13] Luca Bisognin, Thibault Carron, and Jean-Charles Marty. Learning Games Factory: Construction of Learning Games Using a Component-Based Approach. In *Proceedings of the 4th European Conference on Games-Based Learning*, pages 19–30, Reading, UK, 2010. Academic Publishing Limited.
- [14] Boyan Bontchev and Dessislava Vassileva. Courseware Authoring for Adaptive E-learning. In *2009 International Conference on Education Technology and Computer*, pages 176–180. IEEE, 2009. ISBN 9780769536095.

- [15] Michael Booth. The AI Systems of Left 4 Dead. Keynote, Fifth Artificial Intelligence and Interactive Digital Entertainment Conference (AIIDE09), 2009.
- [16] Matthias Bopp. Storytelling as a Motivational Tool in Digital Learning Games. In Theo Hug, editor, *Didactics of Microlearning*, pages 261–279. Waxmann, Münster, 2007.
- [17] Isabel Briggs Myers, Mary H. McCaulley, Naomi L. Quenk, and Allen L. Hammer. *MBTI Manual*. Consulting Psychologists Press, Palo Alto, 3 edition, 1998. ISBN 0891061304.
- [18] Peter Brusilovsky. Adaptive Hypermedia. *User Modeling and User-Adapted Interaction*, 11(1-2):87–110, 2001.
- [19] Tasha Buch and Simon Egenfeldt-Nielsen. The learning effect of "Global Conflicts: Palestine". In *Media@Terra Conference*, 2006.
- [20] Dick C.A. Bulterman and Lynda Hardman. Structured multimedia authoring. *ACM Transactions on Multimedia Computing Communications and Applications*, 1(1):89–109, 2005. ISSN 15516857.
- [21] Daniel Burgos, Pablo Moreno-Ger, José Luis Sierra, Baltasar Fernández-Manjón, and Rob Koper. Authoring game-based adaptive units of learning with IMS Learning Design and <e-Adventure>. *International Journal of Learning Technology*, 3(3):252–268, 2007. ISSN 14778386.
- [22] Joseph Campbell. *The Hero with a Thousand Faces*. Princeton University Press, Princeton/Oxford, 1 edition, 1949.
- [23] Beth Cavallari, John Hedburg, and Barry Harper. Adventure games in education: A review. *Australian Journal of Educational Technology*, 8(2):172–184, 1992.
- [24] Marc Cavazza, Fred Charles, and Steven J. Mead. Character-based interactive storytelling. *IEEE Intelligent Systems*, 17(4):17–24, 2002. ISSN 15411672.
- [25] Marc Cavazza, Jean-Luc Lugin, David Pizzi, and Fred Charles. Madame bovary on the holodeck: immersive interactive storytelling. In *Proceedings of the 15th international conference on Multimedia*, MULTIMEDIA '07, pages 651–660. ACM, 2007. ISBN 9781595937025.
- [26] Marc Cavazza, Ronan Champagnat, and Riccardo Leonardi. The IRIS Network of Excellence: Future Directions in Interactive Storytelling. In Ido A. Iurgel, Nelson Zagalo, and Paolo Petta, editors, *ICIDS '09 Proceedings of the 2nd Joint International Conference on Interactive Digital Storytelling: Interactive Storytelling*, pages 8–13, Berlin / Heidelberg, 2009. Springer.
- [27] Darryl Charles, Michael McNeill, Moira McAlister, Michaela Black, Adrian Moore, Karl Stringer, Julian Kücklich, and Aphra Kerr. Player-Centred Game Design: Player Modelling and Adaptive Digital Games. In *Proceedings of the Digital Games Research Conference*, pages 285–298. DiGRA, 2005.
- [28] Edmund M. Clarke, Orna Grumberg, and Doron Peled. *Model Checking*. MIT Press, 1999. ISBN 0262032708.

- [29] D. S. Cohen and Sergio A. Bustamante II. *Producing Games: From Business and Budgets to Creativity and Design*. Focal Press, 2009. ISBN 0240810708.
- [30] Marcello Coiana, Alex Conconi, Laurence Nigay, and Michael Ortega. Test-Bed for Multimodal Games on Mobile Devices. In Panos Markopoulos, Boris de Ruyter, Wijnand IJsselsteijn, and Duncan Rowland, editors, *Fun and Games*, volume 5294 of *Lecture Notes in Computer Science*, pages 75–87, Berlin / Heidelberg, 2008. Springer. ISBN 9783540883210.
- [31] Owen Conlan and Vincent Wade. Evaluation of APeLS - An Adaptive eLearning Service based on the Multi-model, Metadata-driven Approach. In P. De Bra and W. Nejdl, editors, *Third International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems (AH2004)*, pages 291–295, Berlin / Heidelberg, 2004. Springer.
- [32] Chris Crawford. *Chris Crawford on Interactive Storytelling*. New Riders Games, 2005.
- [33] Mihaly Csikszentmihalyi. *Flow: The Psychology of Optimal Experience*. Harper Perennial, 1991. ISBN 0060920432.
- [34] Martyn Dade-Robertson. Visual Scenario Representation in the Context of a Tool for Interactive Storytelling. In Marc Cavazza and Stephane Donikian, editors, *Virtual Storytelling. Using Virtual Reality Technologies for Storytelling*, volume 4871 of *Lecture Notes in Computer Science*, pages 3–12, Berlin / Heidelberg, 2007. Springer. ISBN 3540770372.
- [35] Angel Del Blanco, Javier Torrente, Eugenio Marchiori, Ivan Martinez-Ortiz, Pablo Moreno-Ger, and Baltasar Fernandez-Manjon. Easing Assessment of Game-based Learning with e-Adventure and LAMS. In *Proceedings of the second ACM international workshop on Multimedia technologies for distance learning*, pages 25–30, New York, NY, USA, 2010. ACM.
- [36] Michele D. Dickey. Game Design Narrative for Learning: Appropriating Adventure Game Design Narrative Devices and Techniques for the Design of Interactive Learning Environments. *Educational Technology Research and Development*, 54(3):245–263, 2006. ISSN 10421629.
- [37] Jean-Paul Doignon and Jean-Claude Falmagne. Spaces for the assessment of knowledge. *International Journal of Man-Machine Studies*, 23(2):175–196, 1985. ISSN 00207373.
- [38] Sean C. Duncan. Minecraft, Beyond Construction and Survival. *Well Played*, 1(1):1–22, 2011.
- [39] Richard Van Eck. Digital Game-Based Learning : It’s Not Just the Digital Natives Who Are Restless. . . . *Educause Review*, 41(2):16–30, 2006. ISSN 15276619.
- [40] Abdulmotaleb El Saddik. *Interactive Multimedia Learning - Shared Reusable Visualization-based Modules*. Springer, Heidelberg, 2001. ISBN 978-3-540-41930-3.

- [41] Casper Van Est, Ronald Poelman, and Rafael Bidarra. High-Level Scenario Editing for Serious Games. In Paul Richard and José Braz, editors, *GRAPP 2011 - Proceedings of the International Conference on Computer Graphics Theory and Applications*, pages 339–346. SciTePress, 2010.
- [42] Rui Figueiredo, Antonio Brisson, Ruth Aylett, and Ana Paiva. Emergent Stories Facilitated: An architecture to generate stories using intelligent synthetic characters. In Ulrike Spierling and Nicolas Szilas, editors, *Interactive Storytelling*, pages 218–229, Berlin/Heidelberg, 2009. Springer.
- [43] Eelke Folmer. Component based Game Development - A solution to escalating costs and expanding deadlines? In Heinz W. Schmidt, Ivica Crnkovic, George T. Heineman, and Judith A. Stafford, editors, *Component-Based Software Engineering*, pages 66–73, Berlin / Heidelberg, 2007. Springer.
- [44] Jonathan G. K. Foss and Alexandra I. Cristea. The next generation authoring adaptive hypermedia: using and evaluating the MOT3.0 and PEAL tools. In *Proceedings of the 21st ACM conference on Hypertext and hypermedia*, pages 83–92, New York, NY, 2010. ACM Press. ISBN 9781450300414.
- [45] Tracy Fullerton. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. Morgan Kaufmann, second edition, 2008. ISBN 0240809742.
- [46] James Paul Gee. *What video games have to teach us about literacy and learning*. Palgrave Macmillan, 2003. ISBN 9781403984531.
- [47] Stefan Göbel, Rainer Malkewitz, and Felicitas Becker. Story Pacing in Interactive Storytelling. In Zhigeng Pan, Ruth Aylett, Holger Diener, Xiaogang Jin, Stefan Göbel, and Li Li, editors, *Technologies for E-Learning and Digital Entertainment*, pages 419–428, Berlin / Heidelberg, 2006. Springer.
- [48] Stefan Göbel, Robert Konrad, Luca Salvatore, Sebastian Sauer, and Kerstin Oswald. U-CREATE: Authoring Tool for the Creation of Interactive Storytelling Based Edutainment Applications. In *Eva 2007 Florence. Electronic Imaging and the Visual Arts : Conference, Training and Exhibition*, pages 53–58, Bologna, January 2007. Pitagora Editrice.
- [49] Stefan Göbel, Viktor Wendel, Christopher Ritter, and Ralf Steinmetz. Personalized, Adaptive Digital Educational Games using Narrative, Game-based Learning Objects. In Xiaopeng Zhang, Shaochun Zhong, Zhigeng Pan, Kevin Wong, and Ruwei Yun, editors, *Entertainment for Education. Digital Techniques and Systems*, pages 438–445, Berlin/Heidelberg, August 2010. Springer. ISBN 978-3-642-14532-2.
- [50] Stefan Göbel, Florian Mehm, and Viktor Wendel. Adaptive Digital Storytelling for Digital Educational Games. In Michael D. Kickmeier-Rust and Dietrich Albert, editors, *An Alien's Guide to Multi-Adaptive Educational Computer Games*, chapter 5, pages 89–104. Informing Science Press, Santa Rosa, USA, 2012. ISBN 1-932886-56-7.
- [51] Gayle H. Gregory. *Differentiating Instruction With Style: Aligning Teacher and Learner Intelligences for Maximum Achievement*. Corwin, 2005. ISBN 0761931627.

- [52] Begoña Gros. Digital Games in Education: The Design of Games-Based Learning Environments. *Journal of Research on Technology in Education*, 1(40):23–38, 2007.
- [53] Casper Harteveld. *Triadic Game Design*. Springer, 2011. ISBN 9781849961561.
- [54] Pei-Chi Ho, Chun-Hsiung Huang, and Szu-Ming Chung. A Computer Adventure Game Applied in E-Learning. In *The 2007 International Conference on Intelligent Pervasive Computing (IPC 2007)*, pages 446–451. IEEE, 2007. ISBN 9780769530062.
- [55] Cord Hockemeyer. Competence Based Adaptive E-Learning in Dynamic Domains. In T. Okamoto, D. Albert, T. Honda, and F. W. Hesse, editors, *The Joint Workshop of Cognition and Learning through Media - Communication for Advanced E-Learning*, pages 79–82, Berlin, 2003. Japanisch-Deutsches Zentrum.
- [56] Paula Hodgson, Donald Man, and Josa Leung. Managing the Development of Digital Educational Games. In Gautam Biswas, Diane Carr, Yam San Chee, and Wu-Yuin Hwang, editors, *International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pages 191–193. IEEE, 2010. ISBN 9781424464333.
- [57] Stefan Hörmann, Tomas Hildebrandt, Christoph Rensing, and Ralf Steinmetz. ResourceCenter - A Digital Learning Object Repository with an Integrated Authoring Tool Set. In Piet Kommers and Griff Richards, editors, *Proceedings of World Conference on Educational Multimedia Hypermedia and Telecommunications EDMEDIA 2005*, pages 3453–3460, Chesapeake, VA, 2005. AACE.
- [58] William Horton and Katherine Horton. *E-learning Tools and Technologies: A consumer's guide for trainers, teachers, educators, and instructional designers*. Wiley, first edition, 2003. ISBN 0471444588.
- [59] Ryan Houlette. Player Modeling for Adaptive Games. In Steve Rabin, editor, *AI Game Programming Wisdom II*, chapter 10, pages 557–566. Charles River Media, Boston, MA, 2004.
- [60] Robin Hunicke. The case for dynamic difficulty adjustment in games. In *Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology*, volume 265 of *ACE '05*, pages 429–433, New York, NY, USA, 2005. ACM. ISBN 1595931104.
- [61] Institute of Electrical and Electronics Engineers, Inc. Ieee standard for learning object metadata, 2002.
- [62] International Standards Organization. Iso 15836-2003 - dublin core metadata element set, version 1.1: Reference description, 2003.
- [63] International Standards Organization. Iso 9241-110:2006 - ergonomics of human-system interaction – part 110: Dialogue principles, 2006.
- [64] Ido Iurgel. *An authoring framework for interactive narrative with virtual characters*. Phd thesis, Technische Universität Darmstadt, 2008.

- [65] Wesley M Johnston, J R Paul Hanna, and Richard J Millar. Advances in dataflow programming languages. *ACM Computing Surveys*, 36(1):1–34, 2004. ISSN 03600300.
- [66] David Keirse and Marilyn Bates. *Please understand me*. Prometheus Nemesis Book Co Inc, first edition, 1984. ISBN 0960695400.
- [67] Caitlin Kelleher. *Motivating programming: using storytelling to make computer programming attractive to middle school girls*. Phd thesis, Carnegie Mellon University, 2006.
- [68] Caitlin Kelleher and Randy Pausch. Lowering the barriers to programming. *ACM Computing Surveys*, 37(2):83–137, 2005. ISSN 03600300.
- [69] Caitlin Kelleher and Randy Pausch. Lessons Learned from Designing a Programming System to Support Middle School Girls Creating Animated Stories. In *IEEE Symposium on Visual Languages and Human-Centric Computing*, pages 165–172. IEEE, 2006. ISBN 0769525865.
- [70] Michael Kickmeier-Rust and Dietrich Albert. Emergent Design: Serendipity in Digital Educational Games. In Randall Shumaker, editor, *Virtual and Mixed Reality*, volume 5622 of *Lecture Notes in Computer Science*, pages 206–215, Berlin / Heidelberg, 2009. Springer.
- [71] Michael Kickmeier-Rust, Daniel Schwarz, Dietrich Albert, Dominique Verpoorten, Jean-Loup Castaigne, and Matthias Bopp. The ELEKTRA project: Towards a new learning experience. In Margit Pohl, Andreas Holzinger, and Renate Motschnig, editors, *M3 - Interdisciplinary Aspects on Digital Media & Education. Proceedings of the 2nd Symposium of the WG HCI & UE of the Austrian Computer Society.*, pages 19–48, 2006.
- [72] Michael Kickmeier-Rust, Stefan Göbel, and Dietrich Albert. 80Days : Melding Adaptive Educational Technology and Adaptive and Interactive Storytelling in Digital Educational Games. In Ralf Klamma, Nalin Sharda, Baltasar Fernández-Manjón, Harald Kosch, and Marc Spaniol, editors, *Proceedings of the First International Workshop on Story-Telling and Educational Games (STEG’08)*. CEUR Workshop Proceedings, January 2008. ISBN ISSN 1613-0073.
- [73] Michael D Kickmeier-Rust and Dietrich Albert, editors. *An Alien’s Guide to Multi-Adaptive Educational Computer Games*. Informing Science Press, Santa Rosa, USA, 2012. ISBN 9781932886566.
- [74] Michael D. Kickmeier-Rust, Neil Peirce, Owen Conlan, Daniel Schwarz, Dominique Verpoorten, and Dietrich Albert. Immersive Digital Games : The Interfaces for Next-Generation E-Learning? In Constantine Stephanidis, editor, *Proceedings of the 4th international conference on Universal access in human-computer interaction: applications and services*, volume 4556 of *Lecture Notes in Computer Science*, pages 647–656, Berlin/Heidelberg, 2007. Springer. ISBN 9783540732822.
- [75] Michael D. Kickmeier-Rust, Cord Hockemeyer, Dietrich Albert, and Thomas Augustin. Micro Adaptive, Non-invasive Knowledge Assessment in Educational Games. In Mike Eisenberg, Kinshuk, Maiga Chang, and Rory McGreal,

- editors, *Second IEEE International Conference on Digital Games and Intelligent Toys Based Education*, volume November 1, pages 135–137. IEEE, 2008. ISBN 9780769534091.
- [76] Annika Kliem and Josef Wiemeyer. Comparison of a Traditional and a Video Game Based Balance Training Program. *International Journal of Computer Science in Sport*, 9(Special Edition Serious Games for Sports and Health):37–50, 2010.
- [77] Annika Kliem, Viktor Wendel, Christian Winter, Josef Wiemeyer, and Stefan Göbel. Virtual Sports Teacher - A Serious Game in Higher Education. In *Serious Games - Theory, Technology & Practice*, pages 61 – 72, 2011.
- [78] Roman Knöll and Mira Mezini. Pegasus: first steps toward a naturalistic programming language. In *Companion to the 21st ACM SIGPLAN symposium on Objectoriented programming systems languages and applications*, pages 542–559, New York, NY, USA, 2006. ACM. ISBN 159593491X.
- [79] Evgeny Knutov, Paul De Bra, and Mykola Pechenizkiy. AH 12 years later: a comprehensive survey of adaptive hypermedia methods and techniques. *The New Review of Hypermedia and Multimedia - Adaptive Hypermedia*, 15(1):5–38, 2009. ISSN 13614568.
- [80] David A. Kolb. *Experiential learning: experience as the source of learning and development*. Prentice Hall, Englewood Cliffs, N.J., 1984.
- [81] Johannes Konert, Kristina Richter, Florian Mehm, Stefan Göbel, Regina Bruder, and Ralf Steinmetz. PEDALE: A Peer Education Diagnostic and Learning Environment. *Journal of Educational Technology & Society*, 15(4):27 – 38, 2012.
- [82] Michael Kriegel and Ruth Aylett. Emergent Narrative as a Novel Framework for Massively Collaborative Authoring. In Helmut Prendinger, James Lester, and Mitsuru Ishizuka, editors, *Proceedings of the 8th international conference on Intelligent Virtual Agents*, pages 73–80, Berlin / Heidelberg, 2008. Springer.
- [83] Robin D. Laws. *Robin's Laws of Good Game Mastering*. Steve Jackson Games, 2001.
- [84] Craig A. Lindley. The Semiotics of Time Structure in Ludic Space As a Foundation for Analysis and Design. *Game Studies*, 5(1), 2005.
- [85] Craig A. Lindley, Lennart Nacke, and Charlotte C. Sennersten. Dissecting Play - Investigating the Cognitive and Emotional Motivations. In *International Conference on Computer Games*, pages 9–16, Wolverhampton, UK, 2008. UNIV WOLVERHAMPTON.
- [86] Ricardo Lopes and Rafael Bidarra. Adaptivity Challenges in Games and Simulations: A Survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 3(2):85–99, 2011. ISSN 1943068X.
- [87] S. Louchart and R.S. Aylett. Narrative Theory and Emergent Interactive Narrative. *International Journal of Continuing Engineering Education and Life-long Learning, special issue on narrative in education*, 14(6):506–518, 2004.



- [88] Sandy Louchart and Ruth Aylett. Managing a Non-linear Scenario - A Narrative Evolution. In Gérard Subsol, editor, *International Conference on Virtual Storytelling*, pages 148–157, Berlin / Heidelberg, 2005. Springer.
- [89] Dennis Maciuszek and Alke Martens. A Reference Architecture for Game-based Intelligent Tutoring. In Patrick Felicia, editor, *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches*, pages 658–682. IGI Global, 2011.
- [90] Dennis Maciuszek, Géraldine Ruddeck, and Alke Martens. Component-based development of educational games: The case of the user interface. In *Proceedings of the 4th European Conference on Games- Based Learning*, pages 208–217, Reading, UK, 2010. Academic Publishing Limited.
- [91] Brian Magerko, John E Laird, Mazin Assanie, Alex Kerfoot, and Devvan Stokes. AI Characters and Directors for Interactive Computer Games. In Randall Hill, editor, *Proceedings of the 16th conference on Innovative applications of artificial intelligence*, volume 1001, pages 877–883. AAAI Press, 2004.
- [92] Brian Magerko, Carrie Heeter, Joe Fitzgerald, and Ben Medler. Intelligent adaptation of digital game-based learning. In Bill Kapralos, Mike Katchabaw, and Jay Rajnovich, editors, *Proceedings of the 2008 Conference on Future Play: Research Play Share Future, Future Play '08*, pages 200–203. ACM Press, 2008. ISBN 9781605582184.
- [93] Steffen Malo and Petra Müsebeck. Winterfest - An Adventure Game for Basic Education. In *3rd workshop on Inclusive E-Learning*, pages 61–65, Hildesheim, 2010. Franzbecker.
- [94] John Maloney, Kylie Peppler, Yasmin B Kafai, Mitchel Resnick, and Natalie Rusk. Programming by choice: urban youth learning programming with scratch. *ACM SIGCSE Bulletin*, 40(1):367–371, 2008. ISSN 00978418.
- [95] Eugenio J. Marchiori, Ángel del Blanco, Javier Torrente, Iván Martínez-Ortiza, and Baltasar Fernández-Manjón. A visual language for the creation of narrative educational games. *Journal of Visual Languages & Computing*, 22(11):443–452, 2011.
- [96] Maic Masuch, Ralf Schmidt, and Kathrin Gerling. Serious Games im Unternehmenskontext: Besonderheiten, Chancen und Herausforderungen der Entwicklung. In Maren Metz and Fabienne Theis, editors, *Digitale Lernwelt - Serious Games: Einsatz in der beruflichen Weiterbildung*, pages 27–38. Bertelsmann, Bielefeld, 2011. ISBN 3763948074.
- [97] Michael Mateas and Andrew Stern. Façade : An Experiment in Building a Fully-Realized Interactive Drama. In *Game Developers Conference (GDC'03)*, volume 2. Georgia Tech, 2003.
- [98] Nick Montfort and Ian Bogost. *Racing the Beam: The Atari Video Computer System (Platform Studies)*. The MIT Press, 2009. ISBN 978-0262012577.

- [99] Pablo Moreno-Ger, Pilar Sancho Thomas, Iván Martínez-Ortiz, José Luis Sierra, and Baltasar Fernández-Manjón. Adaptive Units of Learning and Educational Videogames. *Journal of Interactive Media in Education. Special Issue: Adaptation and IMS Learning Design*, pages 1–15, 2007.
- [100] Pablo Moreno-Ger, Rubén Fuentes-Fernández, José-Luis Sierra-Rodríguez, and Baltasar Fernández-Manjón. Model-checking for adventure videogames. *Information and Software Technology*, 51(3):564–580, 2009. ISSN 09505849.
- [101] Juergen Musil, Angelika Schweda, Dietmar Winkler, and Stefan Biffl. Improving Video Game Development: Facilitating Heterogeneous Team Collaboration Through Flexible Software Processes. In Andreas Riel, Rory O'Connor, Serge Tichkiewitch, and Richard Messnarz, editors, *Systems, Software and Services Process Improvement*, volume 99 of CCIS, pages 83–94. Springer, 2010. ISBN 9783642156656.
- [102] Graham Nelson. Natural Language, Semantic Analysis and Interactive Fiction. Technical report, St. Anne's College, Oxford, 2006.
- [103] Elina M. I. Ollila, Riku Suomela, and Jussi Holopainen. Using prototypes in early pervasive game development. *Computers in Entertainment*, 6(2):149–156, 2008. ISSN 15443574.
- [104] Jon Orwant. EGGG: Automated programming for game generation. *IBM Systems Journal*, 39(3.4):782–794, 2000. ISSN 00188670.
- [105] Mark Overmars. Teaching computer science through game design. *Computer*, 37(4):81–83, 2004. ISSN 00189162.
- [106] Seymour Papert. Does Easy Do It? Children, Games, and Learning. *Game Developer*, 5(June 1998):1–5, 1998.
- [107] Ok-choon Park and Jung Lee. Adaptive Instructional Systems. In Michael J. Spector, M. David Merrill, Jeroen van Merriënboer, and Marcy P. Driscoll, editors, *Handbook of Research on Educational Communications and Technology*, chapter 37, pages 469–484. Taylor Graham, third edition edition, 2008. ISBN 0805841458.
- [108] Federico Peinado, Pablo Gervás, and Álvaro Navarro. A testbed environment for interactive storytellers. In *Proceedings of the 2nd international conference on INtelligent TEchnologies for interactive enterTAINment*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008. ISBN 9789639799134.
- [109] Neil Peirce, Owen Conlan, and Vincent Wade. Adaptive Educational Games: Providing Non-invasive Personalised Learning Experiences. In *Second IEEE International Conference on Digital Game and Intelligent Toy Enhanced Learning*, pages 28–35. IEEE, 2008. ISBN 9780769534091.
- [110] Fabio Petrillo and Marcelo Pimenta. Is agility out there?: agile practices in game development. In *Proceedings of the 28th ACM International Conference on Design of Communication*, pages 9–15, New York, NY, USA, 2010. ACM.

- [111] Andreas Pleuss. MML: a language for modeling interactive multimedia applications. In *Seventh IEEE International Symposium on Multimedia*, Seventh IEEE International Symposium on Multimedia. IEEE Computer Society, 2006.
- [112] Pithamber R. Polsani. Use and Abuse of Reusable Learning Objects. *Journal Of Digital Information*, 3(4):1–11, 2003. ISSN 13687506.
- [113] Marc Prensky. *Digital Game-Based Learning*. McGraw-Hill, 2001.
- [114] Marc Prensky. *Digital Game-Based Learning*. Paragon House, 2007. ISBN 978-1557788634.
- [115] Vladimir Propp. *Morphology of the Folktale*. University of Texas Press, 1968. ISBN 0292783760.
- [116] Jochen Prümper. Der Benutzungsfragebogen ISONORM 9241/10: Ergebnisse zur Reabilität und Validität. In *Software-Ergonomie '97 - Usability Engineering: Integration von Mensch-Computer-Interaktion und Software-Entwicklung*, pages 253–262, Stuttgart, 1997. Teubner.
- [117] Aaron Reed. *Creating Interactive Fiction with Inform 7*. Course Technology PTR, first edition, 2010. ISBN 1435455061.
- [118] Christoph Rensing, Sonja Bergsträßer, Tomas Hildebrandt, Marek Meyer, Birgit Zimmermann, Andreas Faatz, Lasse Lehmann, and Ralf Steinmetz. Re-Use and Re-Authoring of Learning Resources - Definitions and Examples (KOM-TR-2005-02). Technical report, Technische Universität Darmstadt - Multimedia Kommunikation, Darmstadt, 2005.
- [119] Mitchel Resnick and Brian Silverman. Some reflections on designing construction kits for kids. In Mike Eisenberg and Ann Eisenberg, editors, *Proceeding of the 2005 conference on Interaction design and children*, pages 117–122, New York, NY, USA, 2005. ACM Press. ISBN 1595930965.
- [120] Mitchel Resnick, Brian Silverman, Yasmin Kafai, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, and Jay Silver. Scratch: Programming for all. *Communications of the ACM*, 52(11):60–67, November 2009. ISSN 00010782.
- [121] Christian Reuter, Viktor Wendel, Stefan Göbel, and Ralf Steinmetz. Multiplayer Adventures for Collaborative Learning With Serious Games. In Patrick Felicia, editor, *Proceedings of the 6th European Conference on Games Based Learning*, pages 416–423, Reading, UK, 2012. Academic Conferences International Limited.
- [122] Emanuel Montero Reyno and José Á Carsí Cubel. A Platform-Independent Model for Videogame Gameplay Specification. In Atkins Barry, Kennedy Helen, and Krzywinska Tanya, editors, *Breaking New Ground: Innovation in Games, Play, Practice and Theory*. Brunel University, 2009.
- [123] Lloyd P. Rieber. Seriously considering play: Designing interactive learning environments based on the blending of microworlds, simulations, and games. *Educational Technology Research & Development*, 44(2):43–58, 1996. ISSN 10421629.

- [124] Jonathan Rowe, Bradford Mott, Scott McQuiggan, Jennifer Robison, Sunyoung Lee, and James Lester. Crystal Island: A Narrative-Centered Learning Environment for Eighth Grade Microbiology. In Scotty D. Craig, Darina Dicheva, H. Chad Lane, Amy Ogan, and Valerie Shute, editors, *Proceedings of the AIED'09 Workshop on Intelligent Educational Games*, pages 11–20. IOS Press, 2009.
- [125] Winston W. Royce. Managing the development of large software systems. In Sterling M McMurrin, editor, *IEEE WESCON*, pages 1–9. Los Angeles, 1970. ISBN 0897912160.
- [126] Katie Salen and Eric Zimmerman. *Rules of Play: Game Design Fundamentals*. MIT Press, 2003. ISBN 0262240459.
- [127] David Williamson Shaffer. *How computer games help children learn*. Palgrave Macmillan, first edition, 2006. ISBN 1403975051.
- [128] Michael A Shapiro, Jorge Peña Herborn, Jeffrey T Hancock, Peter Vorderer, and Jennings Bryant. Realism, Imagination, and Narrative Video Games. In Peter Vorderer and Jennings Bryant, editors, *Playing Video Games: Motives, Responses and Consequences*, chapter 19, pages 275–289. Lawrence Erlbaum Associates Publishers, 2006. ISBN 9780805853223.
- [129] Manu Sharma, Manish Mehta, Santiago Ontañón, and Ashwin Ram. Player Modeling Evaluation for Interactive Fiction. In *Third Artificial Intelligence for Interactive Digital Entertainment Conference (AIIDE), Workshop on Optimizing Player Satisfaction*, pages 1–18. AAAI Press, 2007.
- [130] Lee Sheldon. *Character Development and Storytelling for Games*. Course Technology PTR, 2004. ISBN 1592003532.
- [131] Valerie Shute and Brendon Towle. Adaptive E-Learning. *Educational Psychologist*, 38(2):105–114, 2003. ISSN 00461520.
- [132] Miguel Sicart. Defining Game Mechanics. *Game Studies*, 8(2):1–14, 2008. ISSN 16047982.
- [133] Raymund Sison and Masamichi Shimura. Student Modeling and Machine Learning. *International Journal of Artificial Intelligence in Education*, 9(1-2):128–158, 1998.
- [134] Gillian Smith, Elaine Gan, Alexei Othenin-girard, and Jim Whitehead. PCG-based game design: enabling new play experiences through procedural content generation. In *Proceedings of the 2nd International Workshop on Procedural Content Generation in Games PCGames 11*. ACM Press, 2011. ISBN 9781450308045.
- [135] Paul Sommeregger and Gudrun Kellner. Brief Guidelines for Educational Adventure Games Creation (EAGC). In Masanori Sugimoto, Vincent Aleven, Yam San Chee, and Baltasar Fernández-Manjón, editors, *International Conference On Digital Game And Intelligent Toy Enhanced Learning*, pages 120–22. IEEE Computer Society, 2012.

- [136] Ulrike Spierling and Nicolas Szilas. Authoring Issues beyond Tools. In Ido A. Iurgel, Nelson Zagalo, and Paolo Petta, editors, *Interactive Storytelling*, pages 50–61, Berlin / Heidelberg, 2009. Springer.
- [137] Ulrike Spierling, Sebastian A. Weiß, and Wolfgang Müller. Towards Accessible Authoring Tools for Interactive Storytelling. In Stefan Göbel, Rainer Malke-witz, and Ido Iurgel, editors, *Technologies for Interactive Digital Storytelling and Entertainment*, volume 4326, pages 169–180, Berlin Heidelberg, 2006. Springer. ISBN 9783540499343.
- [138] Ralf Steinmetz and Klara Nahrstedt. *Multimedia Applications*. Springer, Berlin / Heidelberg, 2004. ISBN 978-3642074103.
- [139] Harold D. Stolovitch and Sivasailam Thiagarajan. *Frame Games*. Educational Technology Publications, Englewood Cliffs, N.J. , 1980. ISBN 0877781443.
- [140] Nicolas Szilas. BEcool: Towards an Author Friendly Behaviour Engine. In Marc Cavazza and Stéphane Donikian, editors, *Proceedings of the 4th international conference on Virtual storytelling: using virtual reality technologies for story-telling*, pages 102–113, Berlin / Heidelberg, 2007. Springer.
- [141] Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Pub, 1997. ISBN 0201178885.
- [142] Richard Tate, Jana Haritatos, and Steve Cole. HopeLab’s Approach to Re-Mission. *International Journal of Learning and Media*, 1(1):29–35, 2009.
- [143] David Thue, Vadim Bulitko, Marcia Spetch, and Eric Wasylishen. Interactive Storytelling : A Player Modelling Approach. In Jonathan Schaeffer and Michael Mateas, editors, *Proceedings of the Third Artificial Intelligence and Interactive Digital Entertainment Conference, Lecture Notes in Computer Science*, pages 43–48. AAAI Press, 2007. ISBN 9783642106422.
- [144] David Thue, Vadim Bulitko, and Marcia Spetch. Player Modeling for Interactive Storytelling: A Practical Approach. In Steve Rabin, editor, *AI Game Programming Wisdom IV*, volume 3, pages 633–646. Charles River Media, Boston, MA, 2008. ISBN 9783642106422.
- [145] David Thue, Vadim Bulitko, and Marcia Spetch. PaSSAGE: A Demonstration of Player Modelling in Interactive Storytelling. In Christian Darken and Michael Mateas, editors, *Proceedings of the Fourth Artificial Intelligence and Interactive Digital Entertainment Conference*, pages 226–227. AAAI Press, 2008.
- [146] David Thue, Vadim Bulitko, and Marcia Spetch. Making Stories Player-Specific: Delayed Authoring in Interactive Storytelling. In Ulrike Spierling and Nicolas Szilas, editors, *Interactive Storytelling*, pages 230–241, Berlin / Heidelberg, 2008. Springer.
- [147] Roberto Tornero, Javier Torrente, Pablo Moreno-Ger, and Baltasar Fernández Manjón. e-Training DS: An Authoring Tool for Integrating Portable Computer Science Games in e-Learning. In Xiangfeng Luo, Marc Spaniol, Lizhe Wang, Qing Li, Wolfgang Nejdl, and Wu Zhang, editors, *Advances in Web-Based Learning*, pages 259–268, Berlin / Heidelberg, 2010. Springer.

- [148] Javier Torrente, Ángel Del Blanco, Guillermo Cañizal, Pablo Moreno-Ger, and Baltasar Fernández-Manjón. <e-Adventure3D>: An Open Source Authoring Environment for 3D Adventure Games in Education. In Masa Inakage and Adrian David Cheok, editors, *Proceedings of the 2008 International Conference on Advances in Computer Entertainment Technology*, pages 191–194, New York, 2008. ACM.
- [149] Javier Torrente, Jose Angel Vallejo-Pinto, Pablo Moreno-Ger, and Baltasar Fernández-Manjón. Introducing Accessibility Features in an Educational Game Authoring Tool: The <e-Adventure> Experience. In *International Conference on Advanced Learning Technologies*, pages 341–343. IEEE, July 2011. ISBN 978-1-61284-209-7.
- [150] Masashi Toyoda, Buntarou Shizuki, Takahashi Shin, Satoshi Matsuoka, and Etsuya Shibayama. Supporting Design Patterns in a Visual Parallel Data-flow Programming Environment. In *1997 IEEE Symposium on Visual Languages*, pages 76–83. IEEE Computer Society, 1997.
- [151] Chi Dung Tran, Sebastián George, and Iza Marfisi-Schottman. EDoS: An Authoring Environment for Serious Games Design Based on Three Models. In *Proceedings of the 4th European Conference on Games- Based Learning (ECGBL 2010)*, pages 393–402, Reading, UK, 2010. Academic Publishing Limited.
- [152] Minh Quang Tran and Robert Biddle. Collaboration in serious game development. In Bill Kapralos, Mike Katchabaw, and Rajnovich Jay, editors, *Proceedings of the 2008 Conference on Future Play: Research, Play, Share, Future Play '08*, pages 49–56, New York, 2008. ACM Press. ISBN 9781605582184.
- [153] Alexey Tsymbal. The problem of concept drift: definitions and related work. Technical report, Trinity College Dublin, Dublin, 2004.
- [154] Maarten W. van Someren, Yvonne F. Barnard, and Jacobijn A.C. Sandberg. *The Think Aloud Method: A practical guide to modelling cognitive processes*. Academic Press, London, 1994. ISBN 0-12-714270-3.
- [155] Christopher Vogler. *The Writer's Journey: Mythic Structure for Storytellers and Screenwriters*. Michael Wiese Productions, California, 1998.
- [156] Viktor Wendel, Maxim Babarinow, Tobias Hoerl, Sergej Kolmogorov, Stefan Göbel, and and Ralf Steinmetz. Woodment: Web-Based Collaborative Multi-player Serious Game. In Xiaopeng Zhang, Shaochun Zhong, Zhigeng Pan, Kevin Wong, and Ruwei Yun, editors, *Transactions on Edutainment IV*, volume 6250, pages 68–78, Berlin / Heidelberg, August 2010. Springer. ISBN 978-3-642-14483-7.
- [157] Thomas Wernbacher, Alexander Pfeiffer, Michael Wagner, and Jörg Hofstätter. Learning by Playing: Can serious games be fun? In Patrick Felicia, editor, *Proceedings of 6th European Conference on Games Based Learning*, pages 534 – 541, Reading, UK, 2012. Academic Conferences Limited.
- [158] Bernard Widrow and Marcian E. Hoff. Adaptive switching circuits. In *Western Electronic Show and Convention, Volume 4*, pages 96–104. Institute of Radio Engineers, 1960.

- [159] Stephen Yang, Brian Smith, and George Graham. Healthy Video Gaming: Oxymoron or Possibility. *Innovate: Journal of Online Education*, 4(4), 2008.
- [160] Nick Yee. Motivations of Play in Online Games. *Journal of CyberPsychology and Behavior*, 9:772–775, 2007.

## LIST OF FIGURES

---

|           |   |    |
|-----------|---|----|
| Figure 1  | The flow channel between boredom and anxiety. ....  | 8  |
| Figure 2  | The user interface of INSCAPE. ....   | 19 |
| Figure 3  | The spectrum of authoring tools and game editors. ....  | 21 |
| Figure 4  | "Ludic Space". ....   | 24 |
| Figure 5  | Story Graphs. ....  | 26 |
| Figure 6  | An example script in the Scratch programming language. ....   | 31 |
| Figure 7  | The typical phases in game development. ....  | 32 |
| Figure 8  | The separation between technical staff, game designers and domain experts. ....   | 34 |
| Figure 9  | Application scenario 1: A single author creating an educational game. ....  | 40 |
| Figure 10 | Application scenario 2: A small team creating a game. ....  | 41 |
| Figure 11 | Application scenario 3: Individuals customizing a game. .   | 41 |
| Figure 12 | Answers to the item "Please rate your computer skills"...   | 42 |
| Figure 13 | Rating of the statement "I am able to create a good, didactically sound educational game" ....                                    | 42 |
| Figure 14 | Rating of the statement "I wish to create educational software with an authoring tool." ....                                      | 43 |
| Figure 15 | Answers to the question "How much time would you be willing to commit to authoring a game played by students for 15 minutes" .... | 43 |
| Figure 16 | An overview of the aspects of the authoring tool concept. ....  | 47 |
| Figure 17 | The structure of an adaptive game. ....   | 49 |
| Figure 18 | An example game structure. ....   | 56 |
| Figure 19 | An example action set. ....   | 58 |
| Figure 20 | The relation between authoring tool, game engine and story engine. ....   | 59 |
| Figure 21 | The cycle of updates and adaptations. ....  | 64 |
| Figure 22 | The Writer's Journey story model applied to a game. ....  | 65 |
| Figure 23 | The combination of a knowledge space and a story structure. ....  | 71 |
| Figure 24 | Scene pools. ....   | 74 |
| Figure 25 | LMS: Relative error (average of 50 generated games). ....   | 76 |
| Figure 26 | PaSSAGE: Relative error (average of 50 generated games). ....   | 76 |
| Figure 27 | LMS: variable target model. ....  | 77 |
| Figure 28 | LMS: relative error ....  | 77 |
| Figure 29 | PaSSAGE: variable target model. ....  | 78 |
| Figure 30 | PaSSAGE: relative error ....  | 78 |
| Figure 31 | An example structural template. ....  | 84 |
| Figure 32 | A simple example of an interaction template. ....   | 85 |
| Figure 33 | The conceptual overview of an interaction template. ....  | 87 |
| Figure 34 | Interaction templates: Interface definition. ....   | 88 |
| Figure 35 | Interaction templates: Authoring implementation. ....   | 89 |



|           |   |     |
|-----------|---|-----|
| Figure 36 | Interaction templates: Player implementation. ....  | 90  |
| Figure 37 | Interaction templates: Documentation. ....  | 91  |
| Figure 38 | Interaction templates: Assistance. ....   | 92  |
| Figure 39 | Comparison of views depending on authoring mode. ....   | 94  |
| Figure 40 | The cyclical workflow provided by iterative authoring. ...                                    | 95  |
| Figure 41 | The model checker as part of the authoring tool. ....   | 97  |
| Figure 42 | The results of the questionnaire of the model checking experiment. ....                       | 101 |
| Figure 43 | The architecture of StoryTec. ....  | 103 |
| Figure 44 | The default user interface of StoryTec. ....  | 104 |
| Figure 45 | The Story Editor. ....  | 105 |
| Figure 46 | The Stage Editor. ....  | 106 |
| Figure 47 | The Objects Browser and the Property Editor. ....   | 107 |
| Figure 48 | The Skill Tree Editor. ....   | 108 |
| Figure 49 | The ActionSet Editor. ....  | 109 |
| Figure 50 | The Condition Editor. ....  | 110 |
| Figure 51 | The initial state of the wizard. ....   | 112 |
| Figure 52 | The second wizard page. ....  | 113 |
| Figure 53 | The third wizard page. ....   | 114 |
| Figure 54 | The final wizard page. ....   | 115 |
| Figure 55 | 3D implementation of an interaction template. ....  | 116 |
| Figure 56 | The authoring workflow. ....  | 118 |
| Figure 57 | The architecture of games authored with StoryTec. ....  | 119 |
| Figure 58 | The overview of a game model encoded in ICML. ....  | 119 |
| Figure 59 | The schema definition of NGLOBs in ICML. ....   | 120 |
| Figure 60 | The schema definition of ActionSets in ICML. ....   | 121 |
| Figure 61 | Cross-Platform development using the StoryTec approach. ....                                  | 122 |
| Figure 62 | The rapid prototyping and analysis tool StoryPlay. ....                                       | 123 |
| Figure 63 | The history visualization of StoryPlay. ....  | 124 |
| Figure 64 | The visualization of narrative context in StoryPlay. ....                                     | 124 |
| Figure 65 | The visualization of gaming context and the player model in StoryPlay. ....                   | 125 |
| Figure 66 | The player skills in textual and graph form in StoryPlay. ....                                | 125 |
| Figure 67 | The average results of the questionnaire aggregated by the seven fields of ISO 9241-10. ....  | 129 |
| Figure 68 | The average results of the questionnaire aggregated by the seven fields of ISO 9241-110. .... | 131 |
| Figure 69 | Comparison of average times spent on the three tasks in the two authoring tools. ....         | 135 |
| Figure 70 | Comparison of task completion and correctness rates. ....                                     | 136 |
| Figure 71 | The three layers of the 80Days architecture. ....   | 164 |
| Figure 72 | The user interface of Bat Cave. ....  | 166 |
| Figure 73 | An interaction template of Geograficus. ....  | 168 |
| Figure 74 | The StoryTec support for Wintermute. ....   | 169 |
| Figure 75 | The architecture of Wintermute's integration into StoryTec. ....                              | 170 |
| Figure 76 | A math exercise in PEDALE. ....   | 171 |
| Figure 77 | A hidden objects game in "Der Chaos-Fluch". ....  | 171 |

|           |  |     |
|-----------|--|-----|
| Figure 78 | The interaction template for mathematics-tasks that involve graphs. ....           | 172 |
| Figure 79 | The TechCraft Stage Editor of StoryTec. ....                                       | 173 |
| Figure 80 | The architecture of TechCraft's integration into StoryTec. .                       | 174 |
| Figure 81 | A physical exercise lesson in Virtual Sports Teacher. ....                         | 175 |
| Figure 82 | The architecture of the StoryTec authoring module for Virtual Sports Teacher. .... | 175 |
| Figure 83 | The flying game aspect of the Motivotion 60+ game. ....                            | 176 |
| Figure 84 | The two domain-specific modes for StoryTec in the context of Motivotion 60+. ....  | 177 |

## LIST OF TABLES

---

|          |  |     |
|----------|--|-----|
| Table 1  | Overview of cited examples for adaptive interactive story-telling systems..... | 13  |
| Table 2  | Overview of cited example adaptive game systems. ....                          | 16  |
| Table 3  | Overview of cited authoring tools.....   | 23  |
| Table 4  | The user requirements drawn from all use cases. ....                           | 46  |
| Table 5  | Adaptation schemes .....   | 61  |
| Table 6  | The Myers-Briggs type indicator.....   | 66  |
| Table 7  | An overview of possible model checks and applicable methods. ....              | 98  |
| Table 8  | Performance data from the pilot study on model checking.                       | 100 |
| Table 9  | Number of times each error type has been found. ....                           | 100 |
| Table 10 | An overview of the implementation of concepts as ICML elements.....            | 117 |

## LIST OF ACRONYMS

---

|         |   |
|---------|---|
| ATML    | Artificial Intelligence Markup Language |
| AH      | Adaptive Hypermedia                     |
| CBSE    | Component-Based Software Engineering    |
| DEG     | Digital Educational Game                |
| DGBL    | Digital Game-based Learning             |
| LMS     | Least Mean Squares                      |
| NGLOB   | Narrative Game-Based Learning Object    |
| CBKST   | Competency-Based Knowledge Space Theory |
| KST     | Knowledge Space Theory                  |
| WYSIWYG | What-you-see-is-what-you-get            |
| ICML    | INSCAPE Communication Markup Language   |



## PROJECTS REALIZED WITH STORYTEC

---

**I**N this appendix, we describe a number of projects and games that have been developed using the game framework described in chapter 7 and that have been authored using the StoryTec authoring tool.

### A.1 STORYTEC OPEN COMMUNITY

StoryTec is open to the public in the form of an open community, with the website [www.storytec.de](http://www.storytec.de)<sup>1</sup> serving as a platform for open community members to register and receive updates on StoryTec. The open community is provided with the authoring tool StoryTec as well as the player application StoryPublish. Along with the software, an extensive tutorial is available for download which features a game that makes use of the major functions of StoryTec.

To this date, the open community has 120 members. An analysis of member interests shows that 78% members are from academia, while 22% are from business or other areas. Members are located in 11 countries (Germany, Austria, Italy, USA, France, Greece, Abu Dhabi, Qatar, Poland, Taiwan and Canada).

The diverse group of open community members includes teacher, students, game developers as well as e-Learning providers.

Workshops that introduce new interested users to StoryTec are offered in the context of the Game Days international conference and science-meets-business event to open community members as well as the public. A first workshop has been offered in the course of the Game Days 2012 and the next workshop is planned in the course of the next Game Days in March 2013.

### A.2 80DAYS

8oDays [72] is a European Union funded research project and the successor to the ELEKTRA project (cf. 2.2.3 on page 15). The research goals of 8oDays were to create an immersive and adaptive 3D educational game that teaches geography to a target group of children aged 12 to 14. This was achieved in two storytelling-based adaptive games: a fully immersive 3D game and a demonstration game specifically created for the purpose of analyzing the adaptive features of the 8oDays architecture.

The game's story involves the alien Feon, who is on a reconnaissance mission to planet Earth. As an extension to ELEKTRA, the character is not only involved in adaptive interventions which change the dialogue, but also has more refined behaviors, such as eye movements and body postures. This is intended to increase para-social interaction, thereby increasing the trust the player develops to the NPC. Since the character is used as a mentor, this increased trust was expected to increase the overall learning and gaming experience [128].

---

<sup>1</sup> Last visited January 24, 2013

The basic framework for both demonstrators is shown in Figure 71. As can be seen, the architecture is based on a game model interpreted in the Story Engine as described in chapter 7. The Story Engine is connected to two engines, a game engine (here: the Open Source 3D Engine “Nebula 2”) in which the game as visible to the player is created and a learning engine for keeping track of the learner model. The **NGLOB** aspects of player, story and learner model are updated from the Story Engine and influence the way in which the game continues.

The adaptive features of ELEKTRA were also extended. In 8oDays, the learning engine is comprised of two components, a Skill Assessment Engine (SAE) and a Motivational Assessment Engine (MAE). The Skill Assessment Engine uses Competency-based Knowledge Space theory to track the skill state of the player and recommends cognitive interventions to the game. Cognitive interventions in 8oDays are focused on learning processes in the learner, an example would be a hint the mentor Feon gives the player to help in the task at hand. The MAE focuses on increasing the motivation of the player. This is achieved by tracking two parameters, attention and confidence, that are used to trigger adaptive interventions of a motivational kind when the engine detects the player losing motivation. As an example, the mentor character could encourage the player to try harder.

This architecture is based on the ALIGN (Adaptive Learning in Games through Non-Invasion) system included both in ELEKTRA [109] and 8oDays.

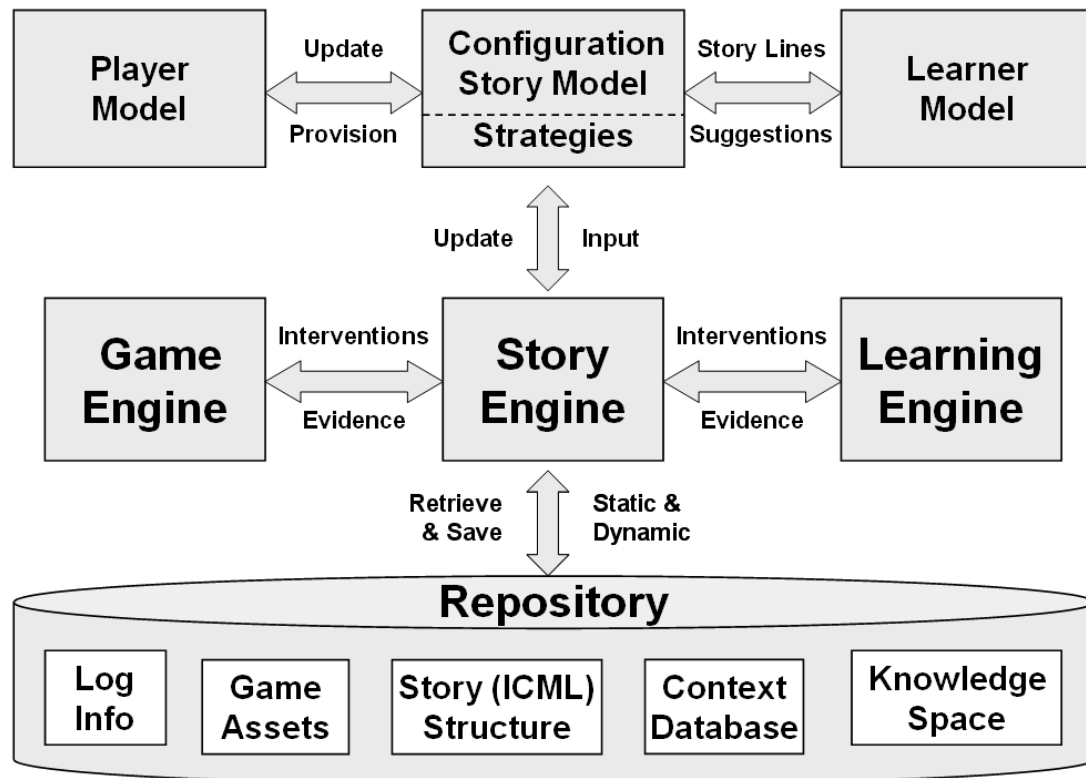


Figure 71: The three layers of the 8oDays architecture (source: [50]).

The two games that were created using the 8oDays architecture are described below.

### A.2.1 *Immersive 3D Demonstrator*

The demonstration game of 80Days is realized with the Nebula 2 game engine, providing a 3D environment for the game. The game throws players into an adventure at the side of an alien (Feon) who was on a mission of cartography for Earth. During the course of this mission, the kid was allowed to fly Feon's spacecraft over a virtual terrain of Europe, solving several tasks along the way.

The concept of "scenes" from the model described in chapter 4 is applied in the form of "micro missions", parts of the game which have a unity of learning content and story elements. The curriculum of the game is drawn from an analysis of curricula of European schools, resulting in a set of learning paths along which the game could be played. Therefore, the micro missions are partially ordered in respect to the prerequisite skills they require. The relationships between the micro missions are therefore modeled as free transitions, allowing an adaptive choice.

Macro adaptivity in 80Days is realized by applying the NGLOB concept to the micro missions. 80Days also supports micro adaptivity in the form of adaptive interventions. These smaller-scale events influence the game inside of a micro mission and can be triggered to give hints when players appear to be unable to solve a task or to give encouragement for players to pursue in their tasks. Examples include changes to texts of characters, extra remarks by characters or ambient changes, for example to increase the tension by playing suspenseful music in the background.

The game itself starts with a login screen used to identify players for evaluation purposes for an overview of the game). This screen realizes a questionnaire asking for information about the players' age, gender, game experience and sensation seeking tendency. The answers to these questions are used to bootstrap the models in the story and learning engine and to set up micro and macro adaptive interventions in the game. The first adaptive choice lies in choosing between one of three major game modes, a "relaxed version", "driven version" and "fast version", which are matched to the players' familiarity with games and tendency for sensation seeking. Differences are manifested for example in time limits, which are not included in the relaxed version but are enforced in the other modes to create tension. This is a realization of the Story Pacing concept, described in section 2.2.2 on page 13.

After the game has started, the story and characters are introduced in an intro sequence. The controls of the game are then explained in a tutorial, followed by a set of micro missions teaching several skills related to the geography of Europe. The structure of the game was designed to follow the Hero's Journey (cf. 5.1 on page 64). For example, the player is taken from his or her ordinary world in the cinematic intro, meets a mentor (Feon) and departs on a heroic journey. The micro missions can be seen as a representation of the "road of trials" stipulated by Campbell.

The 80Days demonstration game was evaluated with 71 participants from secondary schools in Austria and England. The three categories on which the evaluation was based are learning efficacy, usability and user experience. The results indicated that a knowledge gain could be achieved by playing the game (assessed by pre- and posttest questionnaires), and the game as well as the virtual character Feon were well received by the students, indicated by low levels of frustration and a high interest to keep playing the demonstrator game.



### A.2.2 Rapid Prototyping and Evaluation Platform “Bat Cave”

Since it was encountered in the course of the project that testing of the 3D game demonstrator and evaluation of the effects of micro and macro adaptive interventions on the game were complex and time-consuming due to having to play the game each time, a second demonstration in a rapid-prototyping fashion was planned. For this second game, a complete game design document similar to the 80Days game was created, using a similar narrative as the main demonstrator game. This game design also included 132 skills and the knowledge space spanned by them.

This design was fully authored using the StoryTec authoring tool and tested in a rapid prototyping player as described in chapter 6. This application (see also figure 72) was referred to as “Bat Cave”.

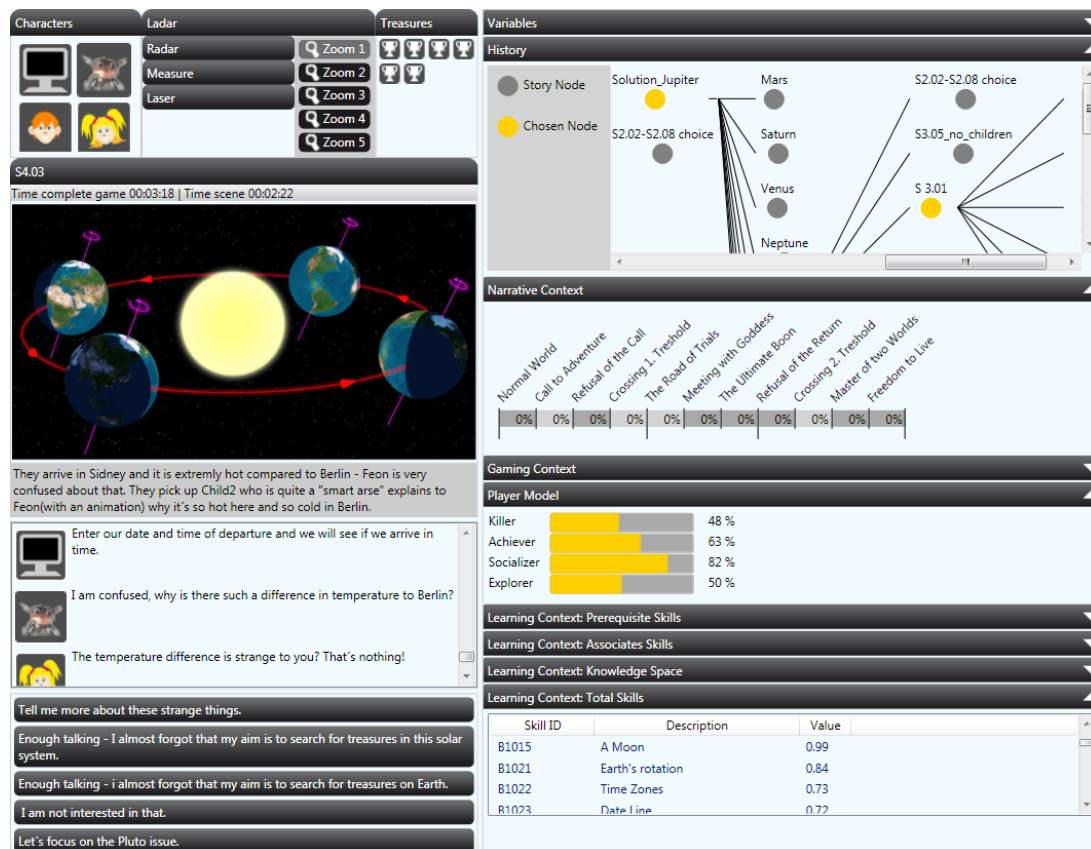


Figure 72: The user interface of Bat Cave.

Even though gameplay is handled by a less immersive 2D user interface than the complete demonstrator game, the architecture as presented in Figure 71 was unchanged and therefore play sessions of Bat Cave are comparable to the full 80Days game.

The Bat Cave game uses a number of mechanisms for input and output. Output to the player is realized by images, descriptive texts, dialogues with characters and treasures (similar to achievements in online games), among others. Input elements are the following:

- Hotspots: By overlaying rectangular, clickable areas over the background image of scenes, a number of effects could be programmed. Each action corresponding to a hotspot is injected into the Story Engine as a stimulus.
- Multiple-Choice Dialogues: Similar to adventure games, the players can communicate with the character Feon by choosing their line of dialogue from a number of alternatives presented at the bottom of the screen, each one linked to a StoryTec stimulus.
- Interaction Templates: Two interaction templates dedicated to this game have been created, in order to allow game mechanics called for in the game design that are of a simulation nature. The templates include heuristics for discretizing the results of the player input into high-level events which are sent to the Story Engine as stimuli. The first template allows players to simulate a flight over Europe and learn the parameters that influence the flight time.

The Bat Cave application furthermore interacts with the Adaptive Learning Engine of 8oDays in order to manage the knowledge state of the player and identify appropriate micro-adaptive interventions. This involves the translation and filtering of stimuli to the relevant game evidence for the learning engine, as well as an interpretation of micro-adaptive interventions into actions (speech acts, help texts and others). These interventions are handled using the StoryTec concepts of stimuli and actions, and therefore require no additional development.

While the process of authoring was not formally tracked, the author tasked with authoring Bat Cave (who was independent of the development of StoryTec) did not encounter serious limitations while authoring the game and was able to structure and fill the game with content in a much shorter timeframe than the full 8oDays game. In preparation, the author received an initial briefing and used the available tutorial scenarios. Furthermore, this author was able to quickly test variations to the game and check the authored parts of the game. The game included 101 scenes and 330 objects.

### A.3 GEOGRAFICUS

The educational adventure game “Geograficus”<sup>2</sup> was originally created by the German educational games developer and publisher Braingame. In a project funded by the Hessian Ministry of Science and Art, this game was used as an example for educational games that could be authored using the authoring tool StoryTec. It was decided to re-author parts of Geograficus and thereby demonstrate the applicability of StoryTec in the context of professional educational game production with real content. Figure 73 displays the StoryTec support for Geograficus using the example of an irregularly shaped puzzle which resembles the continents.

The process of re-authoring was prototypically carried out for a subset of the game that included all relevant game mechanics of the game. The stages that were carried out are the identification of the content to re-author and the identification of requirements for re-authoring.

<sup>2</sup> <http://www.braingame-shop.de/Geograficus-PC-Version-aus-der-Reihe-Clever-Spielen>, last visited on January 24, 2013

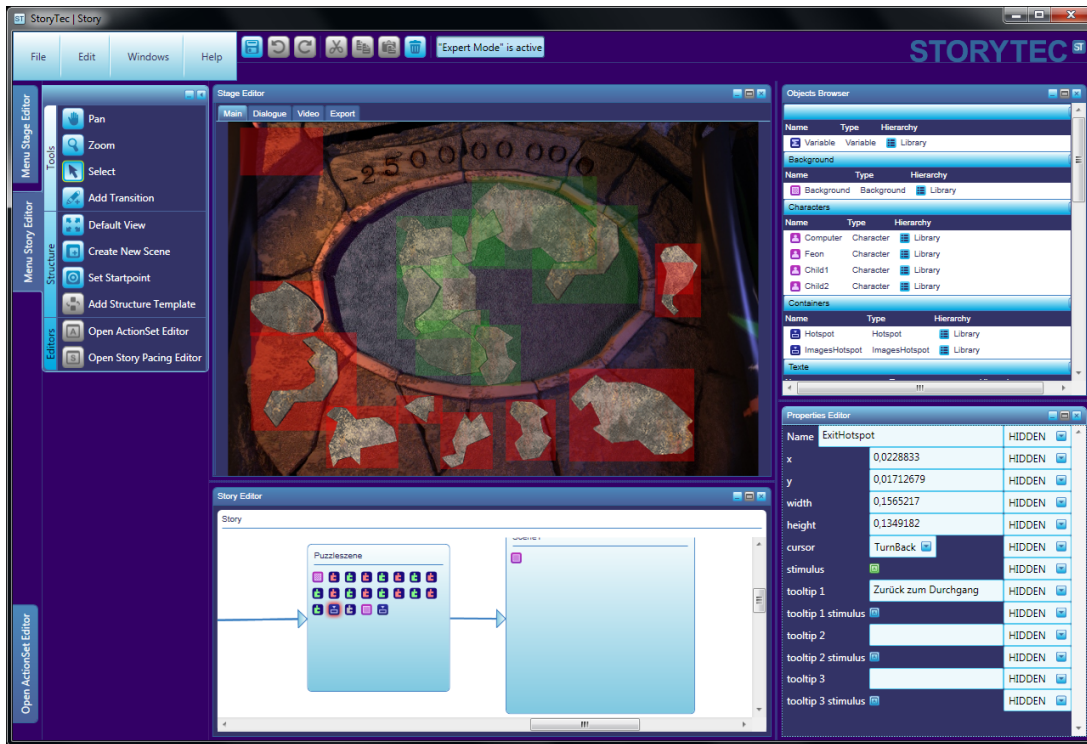


Figure 73: One of the interaction templates for re-authoring Geograficus: an irregularly shaped puzzle.

In this case, since only a part of the game was to be re-authored for demonstrating the viability of the approach, the content was chosen by identifying the relevant game mechanics that are present throughout the game. In a more general re-authoring setting, data from the game's original release (such as player feedback or reviews) could be used to identify parts that should be dropped or that should be extended. The task chosen in our case was that of a puzzle involving a set of gems that have to be arranged in the right order, controlled from a first person perspective in a point and click-fashion.

The identification of requirements encompassed the identification of the game objects and interaction templates that have to be created in order to implement the re-authored game. The main objects identified for Geograficus were images (for showing the virtual world), clickable regions (hotspots) for interaction and navigation and finally a set of interconnected images that are used to model the gems (an interaction that is also used in other areas of the game).

By implementing all necessary interaction templates and structures, including the inventory management found in Geograficus, it could be demonstrated that the complete game could be re-authored using StoryTec.

#### A.4 WINTERMUTE

The Wintermute Engine<sup>3</sup> is a game engine for the creation of adventure games. While it features an editor for such games that is integrated with the game engine, it is not easy for beginners or non-programmers to use. Therefore, an experimental version of

<sup>3</sup> <http://dead-code.org/home/>, last visited on January 24, 2013

StoryTec has been created that targets Wintermute and allows authoring of 2D third person adventure games. This complements the StoryTec version used to re-author the 2D first person adventure game Geograficus as shown in the previous section, thereby allowing the major types of 2D adventure games to be created with StoryTec. Figure 74 is a screenshot of the Wintermute-version of StoryTec.

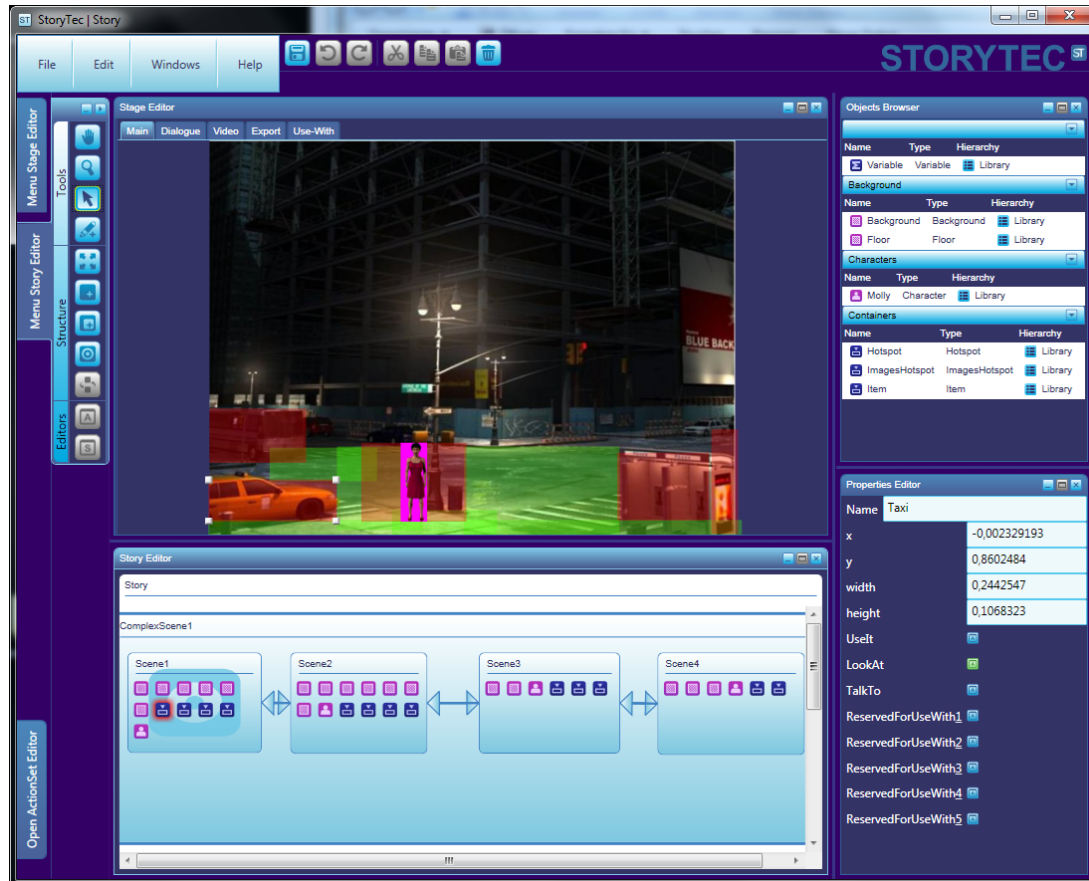


Figure 74: The StoryTec support for Wintermute: A character is placed in a scene with walkable areas and clickable hotspots.

The extension of StoryTec towards Wintermute is centered on the concept of characters that are required to be shown in 2D third person adventure games. In the Wintermute-version of StoryTec, authors can choose from a set of characters that can be placed in a scene and freely moved and scaled to fit them into the perspective of the scene's background image. Furthermore, since the character can be moved by the player via pointing and clicking with the mouse cursor, the walkable areas for the character have to be defined.

In order to allow interaction, hotspots can be added to the scene which define regions that can be clicked by the player. Results of such a click include adding an item to the player's inventory, a dialogue by the character or a change in the scenery, for example a door that opens.

The integration of the Wintermute engine again follows the Story Engine concept as shown in figure 20 on page 59. For the player application, a specialized version of the open-source Wintermute engine that integrates the Story Engine has been created. This version treats some of the game events (such as the player interacting with a hotspot in the scenery) as a stimulus and transmits it to the story engine. In the

other direction, actions are interpreted and forwarded to the appropriate Wintermute subsystems, such as the dialogue system when the author has foreseen a dialogue in response to a stimulus. Figure 80 displays the architecture of this instance of the Story Engine concept.

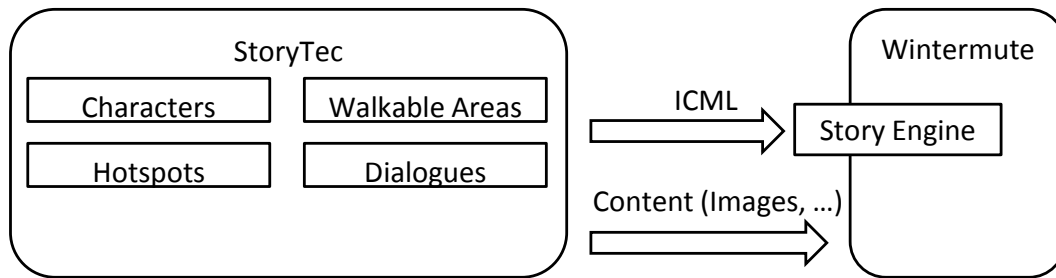


Figure 75: The architecture of Wintermute’s integration into StoryTec

#### A.5 PEDALE

“PEDALE” (Peer Education Diagnostic and Learning Environment)[81] is a learning environment for mathematics that supports social interaction between learners. This social interaction is, on the one hand, intended to allow peer education by letting learners see the approaches of other learners, and to allow peer assessment of open questions on the other hand. The peer assessment aspect applies for all tasks that are not answerable in a computer-readable form. For example, if the task is that of drawing the graph of a specified function, the most intuitive solution of drawing the graph by hand is not machine-readable and therefore not automatically assessable.

In PEDALE, instead, such open tasks are peer-assessed, indicating that the learners switch between solving tasks on their own and then assessing the solutions of others, triggering the meta-cognitive effects of reflection.

PEDALE is implemented on the basis of StoryTec by implementing new interaction templates and integrating the multi-learner aspects required for PEDALE. For the collaborative learning aspect, the interaction template concept has been applied to create the required exercise types for the game. Concerning the peer assessment aspect, several extensions have been created, including a teacher mode in which all learners and their solutions can be managed and a feedback-window in which learners are triggered to give feedback for peers. Underlying these extensions is a database to which all logs and feedbacks are written.

Figure 76 provides a screenshot of the PEDALE-version of StoryTec and of the player for PEDALE content.

#### A.6 DER CHAOS-FLUCH: DARMSTADT IM BANN DES ZAUBERERS

This game was developed in 2011 as a marketing tool for the city of Darmstadt. The narrative casts the player as a marketing official of the city that is charged with finding and correcting the effects of the curse an upset magician has cast on the city. In a first step, the central navigation hub of the game, a map of the city, has to be re-assembled in the form of a puzzle. From this hub, the player can visit four of Darmstadt’s main sights and solve a task at each station. These tasks are realized as



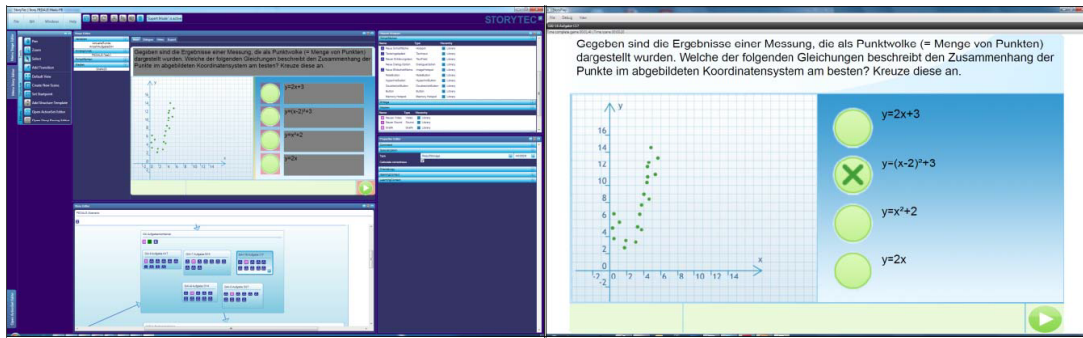


Figure 76: A math exercise in PEDALE (left: authoring in StoryTec, right: the same exercise in the player application; Source: [81])

interaction templates, including a hidden object game, a quiz, a pairs game and a game in which players spot differences between pictures. Figure 77 depicts a scene in the game.



Figure 77: A hidden objects game in “Der Chaos-Fluch”

The game has been published<sup>4</sup> as a HTML5 application running in the browser using the cross-platform publishing mechanism described in section 7.5 on page 118 and has been prominently placed on the city’s web page.

#### A.7 DER WECHSEL

“Der Wechsel”<sup>5</sup> is the title of an educational game for mathematics that has been developed in conjunction with the department of mathematics at Technische Univer-

<sup>4</sup> <http://www.darmstadt.de/portal/darmstadt-aktuell/article/der-chaos-fluch/index.htm>, last visited on January 22, 2013

<sup>5</sup> <http://www3.mathematik.tu-darmstadt.de/ags/didaktik/forschung/didaktik/projekte/fif-seit-2011/der-wechsel-ein-mathekrimi.html>, last visited on January 22, 2013

sität Darmstadt. The game is targeted at students aged 14 to 16 and is centered on changes in representation in mathematical exercises.

In a first step, a script of the whole game was created, combining a crime thriller narrative with mathematical tasks. This game was then implemented as one of the largest games authored using StoryTec. It consists of 258 scenes and 1266 objects. In order to realize the mathematical exercises found in the game, several interaction templates have been designed and implemented. Among them is the interaction template for exercises involving graphs as shown in figure 78.



Figure 78: The interaction template for mathematics-tasks that involve graphs.

## A.8 TECHCRAFT

The game “Minecraft” (cf. [38]) introduced a simple yet popular game mechanic of providing a procedurally generated world made up of blocks to players. Since blocks are easy to destroy and replace, players have been able to create elaborate structures, cities or models, using the game. Importantly, players who are not trained in 3D content creation tools have been able to create large game worlds from their imagination by this simple system. Furthermore, players have created modifications of the game, changing the game rules or adding new objectives.

The observation that the simple in-game interface to the game world has empowered players to become level designers led to a version of StoryTec that incorporates an editing interface similar to Minecraft. For this project, TechCraft<sup>6</sup>, an open source re-implementation of Minecraft has been included into StoryTec by providing a dedicated Stage Editor for block-based editing and a player component. Figure 79 is a screenshot of the user interface of StoryTec with the TechCraft Stage Editor active. In this screenshot, a question block has been placed and configured.

Minecraft’s (and therefore also TechCraft’s) game mechanics of free roaming through a world without a narrative is not the target gameplay of narrative-driven games

<sup>6</sup> <http://techcraft.codeplex.com/>, last visited on January 21, 2013

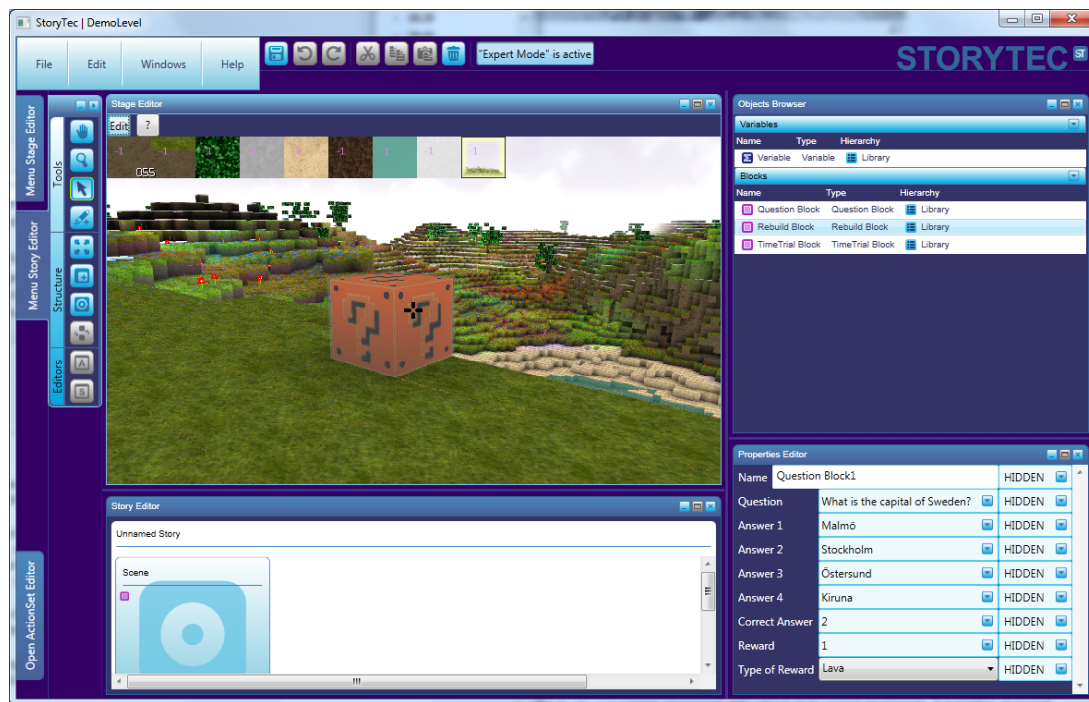


Figure 79: The TechCraft Stage Editor of StoryTec.

that StoryTec has been created for. In Lindley's ludic space (cf. figure 4 on page 24), Minecraft tends towards the simulation and to a lesser extent the game sides than towards the narrative side. Importantly, the game world is not divided spatially, therefore not giving rise to scenes that are defined by their location. Instead, it can be observed that the story structure of a game in this case is a set of quests that players can start whenever they like (cf. figure 5c on page 26).

Authors can create distinct quests for players:

- **Question-Blocks:** In this mode, a block is placed in the world and is associated with a multiple-choice question that players have to answer when they interact with it in the world.
- **Re-Building Tasks:** A structure of blocks (for example, as house) is shown to players in this task. After this presentation, the structure is removed, and players have to re-build it as closely as possible.
- **Time Trials:** In this mode, a race course made of blocks as waypoint markers is shown, and players have to beat the time configured by an author to reach the last waypoint. By the nature of Minecraft's gameplay, waypoints might be initially unreachable and players have to think of ways (e.g. building bridges) to reach them.

In all tasks, authors can specify both the parameters such as the time limit, the questions and answers or the rewards given to players as well as the spatial configuration of the task, for example the placement of the waypoint blocks in a time trial.

Since the block-based interaction with the game world requires the player to control an avatar in the world in order to make movement as intuitive as it is in Minecraft, authors can switch between the normal authoring mode and an in-game authoring



mode. In this mode, the mouse and keyboard input is captured by the 3D part of the Stage Editor and is used to allow players to move around in the game world and manipulate it.

Figure 80 provides an overview of the architecture of the integration of TechCraft into StoryTec. Note that this architecture is another implementation of the concept of a separate Story Engine as shown in figure 20 on page 59.

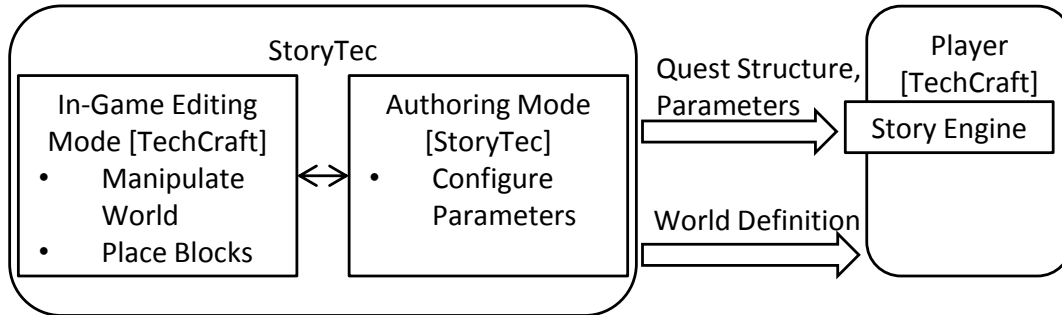


Figure 80: The architecture of TechCraft’s integration into StoryTec.

#### A.9 VIRTUAL SPORTS TEACHER

“Virtual Sports Teacher” [77] is the result of a project funded by the Hessian Ministry of Science and Art. This game is intended as a training environment for teachers, especially in the field of physical education. It follows a script of ideal behavior a sports teacher should exhibit, and places a teacher in a virtual school where he or she can move about freely. The goal is to manage a sports lesson. To this end, the teacher has to assemble all students, check whether their clothing is appropriate, and lead them through the lesson, including the layout of class materials and the correct ways of carrying out the exercises. The screenshot in figure 81 shows the beginning of a game session.

In order to demonstrate the versatility of the project by supplying new lessons, a StoryTec Stage Editor specialized for Virtual Sports Teacher is provided. In this Stage Editor, authors are able to edit two major elements of the game: the teacher’s briefing material and the layout of class materials. The briefing materials contain the knowledge the teacher requires for the lesson in multimedia form. The layout of class materials defines the way in which elements such as mats or racks have to be aligned in order to allow a physical exercise to be executed cleanly and safely.

Figure 82 shows the architecture of the authoring module for Virtual Sports Teacher. Note that in this case, the Story Engine is not used, as the script for the lesson is implemented completely in the game, realized using the Unity game engine. Information is encoded as xml and integrated into the game dynamically.

#### A.10 MOTIVATION 60+

The project “Motivation 60+”, funded by the German Federal Ministry of Education and Research in the field of ambient assisted living, had the goal of providing a Serious Game for elderly people that encouraged physical activity. Accordingly,



Figure 81: A physical exercise lesson in Virtual Sports Teacher.

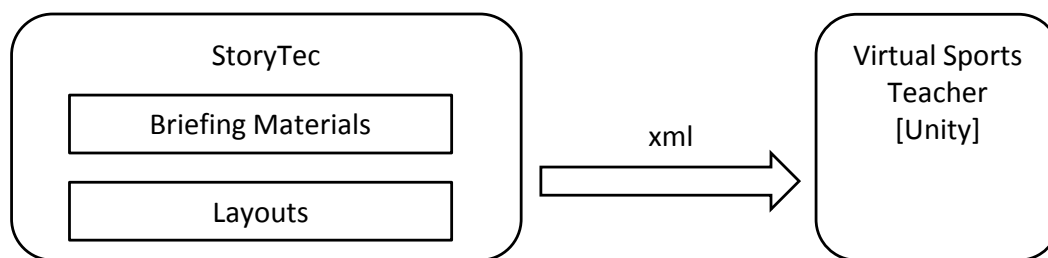


Figure 82: The architecture of the StoryTec authoring module for Virtual Sports Teacher.

this is one example from the field of “games for health”, which is the umbrella term for Serious Games whose purpose is the promotion of the players’ health. In this field, several relevant categories exist. “Exergames” (a portmanteau of “exercise” and “game”) refer to games that incorporate physical movements into the game. Examples include commercial offerings such as Dance Dance Revolution, Wii Fit or EA Sports Active. Other games in this field focus on healthy behavior changes, such as the game presented by Baranowski et al. [7] that trains players to make beneficial nutritional choices.

Games for health are attractive for users due to novel input methods used (often supported by sensors not used in other areas of game interaction) and the playful motivation they provide. Scientific studies have shown the positive effects of games for health; see for example [76] for an account of the effects of exergames on balance training.

Games for health require experts (sports scientists or therapists) both during the development of the game and during the lifetime of the game. By initially contributing to the original design and later by personalizing a game for players, the game

can be adapted to the specific target user groups such as elderly people, persons with disabilities or children.

In the Motivotion 60+ system, two separate activities are offered to the target users:

- Outdoor activities, where players are tracked while exercising outdoors
- Indoor activities, which are realized as a Serious Game that involves body tracking as an input mechanism

Since the target of StoryTec in the context of Motivotion 60+ was the indoor Serious Game, this game is described in detail here. The indoor game is framed as a journey around the world, in which players have to travel between places by emulating the flight of an airplane (see figure 83). Simultaneously, they train their stability and strength by stretching their arms and rotating their upper body. After reaching their destination, the players accomplish one of a set of bodily challenges (for example, doing knee lifts while climbing the stairs of Notre Dame in Paris). Afterwards, they are given information about the target city and are asked a question. This second aspect is done to complement the physical exercise of the game with cognitive stimulation.



Figure 83: The flying game aspect of the Motivotion 60+ game.

The authoring tool StoryTec was used in two aspects of the development of Motivotion 60+. In a first step, the rapid prototyping possibilities of StoryTec were used during the initial design phase to quickly create a game prototype which allowed to explore the journey around the world-theme and the choice of places.

In a second step, the integration of content had to be realized. Here, two main user groups were involved: First, therapists were needed to set up the correct parameters for the exercises in the game, fine-tuning them to the needs of the players. Second, content providers were needed, mainly for providing the content for the cognitive

challenges of motivation 60+ (images and quiz texts). The diagram in figure 84 shows how two different domain-specific views were created in StoryTec as described in section 6.2 on collaborative authoring. Since the game platform (“mbox”) of motivation 60+ was not set up to use the StoryTec runtime architecture, data had to be exported from StoryTec in specific formats as required for the runtime platform. This is an example where the approach of StoryTec can be used to configure applications that are not built to be inherently compatible to it.

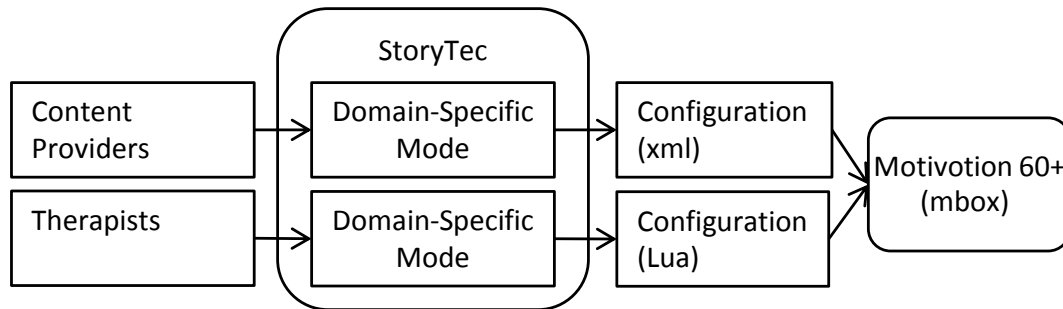


Figure 84: The two domain-specific modes for StoryTec in the context of Motivation 60+.

A further motivation for applying StoryTec in this scenario is the realization that a major shortcoming of exergames is the lack of long-term motivation. Studies such as [159] yielded the result that after a game has been started, motivation is high, but decreases with use. Factors in this decrease are the building knowledge of players about the tasks in the game that, in turn, leads to a less challenging game. By allowing an author to create new challenges for players, the motivation experienced by players can be prolonged. Furthermore, user-generated content (such as exercises or challenges created by friends) are possible with this approach.



QUESTIONNAIRES

---

**I**N this appendix, the questionnaires and task descriptions for the evaluation studies carried out with StoryTec can be found.

The questionnaire items and texts are drawn from the questionnaire “ISONORM 9241/10: Beurteilung von Software auf Grundlage der Internationalen Ergonomie-Norm ISO 9241/10” (Jochen Prümper and Michael Anft, FHTW-Berlin, 1993)<sup>1</sup>.

---

<sup>1</sup> available at [http://www.ergo-online.de/site.aspx?url=html/software/verfahren\\_zur\\_beurteilung\\_der/fragebogen\\_isonorm\\_online.htm](http://www.ergo-online.de/site.aspx?url=html/software/verfahren_zur_beurteilung_der/fragebogen_isonorm_online.htm), last visited on January 24, 2013

## B.1 INITIAL USABILITY STUDY QUESTIONNAIRE

## Fragebogen zum Autorenwerkzeug StoryTec

Bitte füllen Sie den Fragebogen vollständig aus und lassen Sie keine Felder aus. Es gibt kein Richtig oder Falsch. Sollte es Ihnen schwer fallen eine Einschätzung zu geben, wählen Sie bitte die Antwortalternative, die Ihre Erfahrung am besten beschreibt.

Ihre Daten werden selbstverständlich anonym und vertraulich behandelt.

Vielen Dank für Ihre Hilfe!

## Das Autorenwerkzeug StoryTec ...

|     |  | sehr                     | ziemlich                 | eher                     | eher                     | ziemlich                 | sehr                     |  |
|-----|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 1.1 | ist kompliziert zu bedienen.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist unkompliziert zu bedienen.   |
| 1.2 | bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.                                       | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.   |
| 1.3 | bietet schlechte Möglichkeiten, häufig sich wiederholende Bearbeitungsvorgänge zu automatisieren.                        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet gute Möglichkeiten, häufig sich wiederholende Bearbeitungsvorgänge zu automatisieren.                           |
| 1.4 | erfordert überflüssige Eingaben  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert keine überflüssigen Eingaben   |
| 1.5 | ist schlecht auf die Anforderungen der Arbeit zugeschnitten.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist gut auf die Anforderungen der Arbeit zugeschnitten.  |
| 2.1 | bietet einen schlechten Überblick über sein Funktionsangebot.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet einen guten Überblick über sein Funktionsangebot.   |
| 2.2 | verwendet schlecht verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.                  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs.                     |
| 2.3 | liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.                          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | liefert in ausreichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.                         |
| 2.4 | bietet auf Verlangen keine situationsspezifischen Erklärungen, die konkret weiterhelfen.                                 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet auf Verlangen situationsspezifische Erklärungen, die konkret weiterhelfen.                                      |
| 3.1 | bietet keine Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet die Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen. |

|     |  | sehr                     | ziemlich                 | eher                     | eher                     | ziemlich                 | sehr                     |  |
|-----|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 3.2 | erzwingt eine unnötig starre Einhaltung von Bearbeitungsschritten.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erzwingt keine unnötig starre Einhaltung von Bearbeitungsschritten.  |
| 3.3 | ermöglicht keinen leichten Wechsel zwischen einzelnen Menüs oder Masken.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ermöglicht einen leichten Wechsel zwischen einzelnen Menüs oder Masken.  |
| 3.4 | ist so gestaltet, dass der Benutzer nicht beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass der Benutzer beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden. |
| 3.5 | erzwingt unnötige Unterbrechungen der Arbeit.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erzwingt keine unnötigen Unterbrechungen der Arbeit.   |
| 4.1 | erschwert die Orientierung, durch eine uneinheitliche Gestaltung.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erleichtert die Orientierung, durch eine einheitliche Gestaltung.  |
| 4.2 | lässt einen im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt einen nicht im unklaren darüber, ob eine Eingabe erfolgreich war oder nicht.                                   |
| 4.3 | informiert in unzureichendem Maße über das, was es gerade macht.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | informiert in ausreichendem Maße über das, was es gerade macht.  |
| 4.4 | reagiert mit schwer vorhersehbaren Bearbeitungszeiten.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | reagiert mit gut vorhersehbaren Bearbeitungszeiten.  |
| 4.5 | lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.  |
| 5.1 | ist so gestaltet, dass kleine Fehler schwerwiegende Folgen haben können.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass kleine Fehler keine schwerwiegenden Folgen haben können.                                      |
| 5.2 | informiert zu spät über fehlerhafte Eingaben.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | informiert sofort über fehlerhafte Eingaben.   |
| 5.3 | liefert schlecht verständliche Fehlermeldungen.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | liefert gut verständliche Fehlermeldungen.   |
| 5.4 | erfordert bei Fehlern im Großen und Ganzen einen hohen Korrekturaufwand.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert bei Fehlern im Großen und Ganzen einen geringen Korrekturaufwand.  |
| 5.5 | gibt keine konkreten Hinweise zur Fehlerbehebung.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | gibt konkrete Hinweise zur Fehlerbehebung.   |
| 6.1 | lässt sich vom Benutzer schwer erweitern, wenn für ihn neue Aufgaben entstehen.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich vom Benutzer leicht erweitern, wenn für ihn neue Aufgaben entstehen.                                      |



## B.2 FOLLOW-UP USABILITY STUDY QUESTIONNAIRE



## Fragebogen zur Beurteilung des Autorenwerkzeugs StoryTec

Im Folgenden geht es um die Beurteilung des Autorenwerkzeugs StoryTec auf Grundlage der Internationalen Norm ISO 9241/10.

Ziel dieser Beurteilung ist es, Schwachstellen der Software aufzudecken und konkrete Verbesserungsvorschläge zu entwickeln.

Um dies zu bewerkstelligen, ist Ihr Urteil als Tester des Softwaresystems von entscheidender Bedeutung! Grundlage Ihrer Bewertung sind Ihre individuellen Erfahrungen mit StoryTec.

Dabei geht es nicht um eine Beurteilung Ihrer Person, sondern um Ihre persönliche Bewertung der Software.

**Bitte füllen Sie den Fragebogen vollständig aus und lassen Sie keine Felder aus. Es gibt keine richtige oder falsche Antwort. Sollte es Ihnen schwer fallen eine Einschätzung zu geben, wählen Sie bitte die Antwortalternative, die Ihre Erfahrung am besten beschreibt.**

Ihre Daten werden selbstverständlich anonym und vertraulich behandelt.

Vielen Dank für Ihre Hilfe!

| Aufgabenangemessenheit  |                          |                          |                          |                          |                          |                          |                          |  |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| Unterstützt die Software die Erledigungen Ihrer Arbeitsaufgaben, ohne Sie als Benutzer unnötig zu belasten? |                          |                          |                          |                          |                          |                          |                          |  |
| Die Software ...  | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
| 1.1 ist kompliziert zu bedienen.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist unkompliziert zu bedienen.   |
| 1.2 bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.                      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.                 |
| 1.3 bietet schlechte Möglichkeiten, häufig sich wiederholende Bearbeitungsvorgänge zu automatisieren.       | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet gute Möglichkeiten, häufig sich wiederholende Bearbeitungsvorgänge zu automatisieren. |
| 1.4 erfordert überflüssige Eingaben   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert keine überflüssigen Eingaben   |
| 1.5 ist schlecht auf die Anforderungen der Arbeit zugeschnitten.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist gut auf die Anforderungen der Arbeit zugeschnitten.                                      |

## Selbstbeschreibungsfähigkeit

Gibt Ihnen die Software genügend Erläuterungen und ist sie in ausreichendem Maße verständlich?

| Die Software ...  | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 2.1 bietet einen schlechten Überblick über sein Funktionsangebot.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet einen guten Überblick über sein Funktionsangebot.   |
| 2.2 verwendet schlecht verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | verwendet gut verständliche Begriffe, Bezeichnungen, Abkürzungen oder Symbole in Masken und Menüs. |
| 2.3 liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.         | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | liefert in ausreichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind.     |
| 2.4 bietet auf Verlangen keine situationsspezifischen Erklärungen, die konkret weiterhelfen.                | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet auf Verlangen situationsspezifische Erklärungen, die konkret weiterhelfen.                  |
| 2.5 bietet von sich aus keine situationsspezifischen Erklärungen, die konkret weiterhelfen.                 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet von sich aus situationsspezifische Erklärungen, die konkret weiterhelfen.                   |

## Steuerbarkeit

Können Sie als Benutzer die Art und Weise, wie Sie mit der Software arbeiten, beeinflussen?

| Die Software ...   | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 3.1 bietet keine Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet die Möglichkeit, die Arbeit an jedem Punkt zu unterbrechen und dort später ohne Verluste wieder weiterzumachen. |
| 3.2 erzwingt eine unnötige starre Einhaltung von Bearbeitungsschritten   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erzwingt keine unnötige starre Einhaltung von Bearbeitungsschritten  |
| 3.3 ermöglicht keinen leichten Wechsel zwischen einzelnen Menüs oder Masken.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ermöglicht einen leichten Wechsel zwischen einzelnen Menüs oder Masken.  |
| 3.4 ist so gestaltet, dass der Benutzer nicht beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass der Benutzer beeinflussen kann, wie und welche Informationen am Bildschirm dargeboten werden.   |
| 3.5 erzwingt unnötige Unterbrechungen der Arbeit.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erzwingt keine unnötigen Unterbrechungen der Arbeit.   |

## Erwartungskonformität

Kommt die Software durch eine einheitliche und verständliche Gestaltung Ihren Erwartungen und Gewohnheiten entgegen?

| Die Software ...   | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 4.1 erschwert die Orientierung, durch eine uneinheitliche Gestaltung.            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erleichtert die Orientierung, durch eine einheitliche Gestaltung.                  |
| 4.2 lässt einen im Unklaren darüber, ob eine Eingabe erfolgreich war oder nicht. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt einen nicht im unklaren darüber, ob eine Eingabe erfolgreich war oder nicht. |
| 4.3 informiert in unzureichendem Maße über das, was es gerade macht.             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | informiert in ausreichendem Maße über das, was es gerade macht.                    |
| 4.4 reagiert mit schwer vorhersehbaren Bearbeitungszeiten.                       | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | reagiert mit gut vorhersehbaren Bearbeitungszeiten.                                |
| 4.5 lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.                  |

## Fehlertoleranz

Bietet Ihnen die Software die Möglichkeit, trotz fehlerhafter Eingaben das beabsichtigte Arbeitsergebnis ohne oder mit geringem Korrekturaufwand zu erreichen?

| Die Software ...   | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...  |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|---|
| 5.1 ist so gestaltet, dass kleine Fehler schwerwiegende Folgen haben können. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass kleine Fehler keine schwerwiegenden Folgen haben können. |
| 5.2 informiert zu spät über fehlerhafte Eingaben.                            | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | informiert sofort über fehlerhafte Eingaben.                                    |
| 5.3 liefert schlecht verständliche Fehlermeldungen.                          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | liefert gut verständliche Fehlermeldungen.                                      |
| 5.4 erfordert bei Fehlern im Großen und Ganzen einen hohen Korrekturaufwand. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert bei Fehlern im Großen und Ganzen einen geringen Korrekturaufwand.     |
| 5.5 gibt keine konkreten Hinweise zur Fehlerbehebung.                        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | gibt konkrete Hinweise zur Fehlerbehebung.                                      |

## Individualisierbarkeit

Können Sie als Benutzer die Software ohne großen Aufwand auf Ihre individuellen Bedürfnisse und Anforderungen anpassen?

| Die Software ...  | - - -                    | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...  |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|---|
| 6.1 lässt sich vom Benutzer schwer erweitern, wenn für ihn neue Aufgaben entstehen.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich vom Benutzer leicht erweitern, wenn für ihn neue Aufgaben entstehen.   |
| 6.2 lässt sich vom Benutzer schlecht an seine persönliche, individuelle Art der Arbeitserledigung anpassen.                           | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich vom Benutzer gut an seine persönliche, individuelle Art der Arbeitserledigung anpassen.                      |
| 6.3 eignet sich für Anfänger und Experten nicht gleichermaßen, weil der Benutzer es nur schwer an seinen Kenntnisstand anpassen kann. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | eignet sich für Anfänger und Experten gleichermaßen, weil der Benutzer es leicht an seinen Kenntnisstand anpassen kann. |
| 6.4 lässt sich - im Rahmen seines Leistungsumfangs – vom Benutzer schlecht für unterschiedliche Aufgaben passend einrichten.          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich - im Rahmen seines Leistungsumfangs - Benutzer gut für unterschiedliche Aufgaben passend einrichten.         |
| 6.5 ist so gestaltet, dass der Benutzer die Bildschirmdarstellung schlecht an seine individuellen Bedürfnisse anpassen kann.          | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass der Benutzer die Bildschirmdarstellung gut an seine individuellen Bedürfnisse anpassen kann.     |

## Lernförderlichkeit

Ist die Software so gestaltet, dass Sie sich ohne großen Aufwand in sie einarbeiten konnten und bietet sie auch dann Unterstützung, wenn Sie neue Funktionen erlernen möchten?

| Die Software ...  | - - -                    | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 7.1 erfordert viel Zeit zum Erlernen.                               | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert wenig Zeit zum Erlernen.                         |
| 7.2 ermutigt nicht dazu, auch mal neue Funktionen auszuprobieren.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ermutigt dazu, auch mal neue Funktionen auszuprobieren.    |
| 7.3 erfordert, dass man sich viele Details merken muss.             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert nicht, dass man sich viele Details merken muss.  |
| 7.4 ist so gestaltet, dass sich einmal Gelerntes schlecht einprägt. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass sich einmal Gelerntes gut einprägt. |
| 7.5 ist schlecht ohne fremde Hilfe oder Handbuch erlernbar.         | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist gut ohne fremde Hilfe oder Handbuch erlernbar.         |

## Zum Schluss

| Wie gut beherrschen Sie StoryTec? | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      |          |
|-----------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------|
| sehr schlecht                     | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | sehr gut |

| Haben Sie bereits Erfahrungen mit anderen Autorenwerkzeugen gesammelt? | Ja <input type="checkbox"/> | Nein <input type="checkbox"/> | Name der Software:       |                          |                          |                          |                          |          |
|--|-----------------------------|-------------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------|
| Wie gut beherrschen Sie diese Software?                                | ---                         | --                            | -                        | -/+                      | +                        | ++                       | +++                      |          |
| sehr schlecht  | <input type="checkbox"/>    | <input type="checkbox"/>      | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | sehr gut |

| Wie lange arbeiten Sie (privat/beruflich) überhaupt schon mit ...   |                          |
|---|--------------------------|
| Computern?  | _____ Jahre _____ Monate |
| anderen Autorenwerkzeugen?  | _____ Jahre _____ Monate |
| Wie häufig arbeiten Sie (privat/beruflich) durchschnittlich mit ... |                          |
| Computern?  | _____ Stunden pro Woche  |
| anderen Autorenwerkzeugen?  | _____ Stunden pro Woche  |

|                   |   |
|-------------------|---|
| Ihr Geschlecht?   | <input type="checkbox"/> weiblich <input type="checkbox"/> männlich |
| Wie alt sind Sie? | _____ Jahre   |

Vielen Dank für Ihre aktive Hilfe. Ihre Daten werden selbstverständlich anonym behandelt.

Das letzte Wort haben Sie:

Bitte sehen Sie den Fragebogen nochmals durch, ob Sie alle Fragen beantwortet haben!

## B.3 COMPARISON STUDY QUESTIONNAIRE



## Fragebogen zur Beurteilung der Autorenwerkzeuge e-Adventure und StoryTec

Im Folgenden geht es um die Beurteilung der Autorenwerkzeuge e-Adventure und StoryTec.

Ziel dieser Beurteilung ist es, Schwachstellen der Software aufzudecken und konkrete Verbesserungsvorschläge zu entwickeln.

Um dies zu bewerkstelligen, ist Ihr Urteil als Tester des Softwaresystems von entscheidender Bedeutung! Grundlage Ihrer Bewertung sind Ihre individuellen Erfahrungen mit d.

Dabei geht es nicht um eine Beurteilung Ihrer Person, sondern um Ihre persönliche Bewertung der Software.

**Bitte füllen Sie den Fragebogen vollständig aus und lassen Sie keine Felder aus. Es gibt keine richtige oder falsche Antwort. Sollte es Ihnen schwer fallen eine Einschätzung zu geben, wählen Sie bitte die Antwortalternative, die Ihre Erfahrung am besten beschreibt.**

Ihre Daten werden selbstverständlich anonym und vertraulich behandelt.

Vielen Dank für Ihre Hilfe!

| E-Adventure   |                          |                          |                          |                          |                          |                          |                          |  |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| Bewerten Sie das Tool E-Adventure auf Basis Ihrer Erfahrungen                                       |                          |                          |                          |                          |                          |                          |                          |  |
| Die Software ...  | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
| 1.1 ist kompliziert zu bedienen.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist unkompliziert zu bedienen.   |
| 1.2 bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.              | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.                   |
| 1.3 ist schlecht auf die Anforderungen der Arbeit zugeschnitten.                                    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist gut auf die Anforderungen der Arbeit zugeschnitten.  |
| 1.4 bietet einen schlechten Überblick über sein Funktionsangebot.                                   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet einen guten Überblick über sein Funktionsangebot.                                       |
| 1.5 liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | liefert in ausreichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind. |
| 1.6 erzwingt eine unnötige starre Einhaltung von Bearbeitungsschritten                              | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erzwingt keine unnötige starre Einhaltung von Bearbeitungsschritten                            |

|          |   |                          |                          |                          |                          |                          |                          |                          |   |
|----------|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|---|
| 1.7      | erschwert die Orientierung, durch eine uneinheitliche Gestaltung.               | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erleichtert die Orientierung, durch eine einheitliche Gestaltung.               |
| 1.8      | lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.         | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.               |
| 1.9      | ist so gestaltet, dass kleine Fehler schwerwiegende Folgen haben können.        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass kleine Fehler keine schwerwiegenden Folgen haben können. |
| 1.1<br>0 | erfordert bei Fehlern im Großen und Ganzen einen hohen Korrekturaufwand.        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert bei Fehlern im Großen und Ganzen einen geringen Korrekturaufwand.     |
| 1.1<br>1 | lässt sich vom Benutzer schwer erweitern, wenn für Ihn neue Aufgaben entstehen. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich vom Benutzer leicht erweitern, wenn für Ihn neue Aufgaben entstehen. |
| 1.1<br>2 | erfordert viel Zeit zum Erlernen.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert wenig Zeit zum Erlernen.  |
| 1.1<br>3 | erfordert, dass man sich viele Details merken muss.                             | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert nicht, dass man sich viele Details merken muss.                       |

|   |                          |                          |                          |                          |                          |                          |                          |          |  |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------|--|
|   |                          |                          |                          |                          |                          |                          |                          |          |  |
| <b>Wie gut beherrschen Sie E-Adventure?</b>   | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      |          |  |
| sehr schlecht   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | sehr gut |  |
| Schätzen Sie, wie viele Stunden Einarbeitungszeit Sie benötigen würden um das Tool E-Adventure komplett benutzen zu können. |                          |                          |                          |                          |                          |                          |                          |          |  |
| _____ Stunden   |                          |                          |                          |                          |                          |                          |                          |          |  |

## StoryTec

Bewerten Sie das Tool StoryTec auf Basis Ihrer Erfahrungen

| Die Software ...  | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      | Die Software ...   |
|---|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--|
| 1.1 ist kompliziert zu bedienen.  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist unkompliziert zu bedienen.   |
| 1.2 bietet nicht alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.              | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet alle Funktionen, um die anfallenden Aufgaben effizient zu bewältigen.                   |
| 1.3 ist schlecht auf die Anforderungen der Arbeit zugeschnitten.                                    | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist gut auf die Anforderungen der Arbeit zugeschnitten.  |
| 1.4 bietet einen schlechten Überblick über sein Funktionsangebot.                                   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | bietet einen guten Überblick über sein Funktionsangebot.                                       |
| 1.5 liefert in unzureichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind. | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | liefert in ausreichendem Maße Informationen darüber, welche Eingaben zulässig oder nötig sind. |
| 1.6 erzwingt eine unnötige starre Einhaltung von Bearbeitungsschritten                              | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erzwingt keine unnötige starre Einhaltung von Bearbeitungsschritten                            |
| 1.7 erschwert die Orientierung, durch eine uneinheitliche Gestaltung.                               | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erleichtert die Orientierung, durch eine einheitliche Gestaltung.                              |
| 1.8 lässt sich nicht durchgehend nach einem einheitlichen Prinzip bedienen.                         | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich durchgehend nach einem einheitlichen Prinzip bedienen.                              |
| 1.9 ist so gestaltet, dass kleine Fehler schwerwiegende Folgen haben können.                        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | ist so gestaltet, dass kleine Fehler keine schwerwiegenden Folgen haben können.                |
| 1.1 erfordert bei Fehlern im Großen und Ganzen einen hohen Korrekturaufwand.                        | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert bei Fehlern im Großen und Ganzen einen geringen Korrekturaufwand.                    |
| 1.1 lässt sich vom Benutzer schwer erweitern, wenn für Ihn neue Aufgaben entstehen.                 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | lässt sich vom Benutzer leicht erweitern, wenn für Ihn neue Aufgaben entstehen.                |
| 1.1 erfordert viel Zeit zum Erlernen.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert wenig Zeit zum Erlernen.   |
| 1.1 erfordert, dass man sich viele Details merken muss.   | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | erfordert nicht, dass man sich viele Details merken muss.                                      |
|   |                          |                          |                          |                          |                          |                          |                          |  |
| Wie gut beherrschen Sie StoryTec?   | ---                      | --                       | -                        | -/+                      | +                        | ++                       | +++                      |  |



|  |                          |                          |                          |                          |                          |                          |                          |          |
|--|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|----------|
| sehr schlecht  | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | sehr gut |
| Schätzen Sie, wie viele Stunden Einarbeitungszeit Sie benötigen würden um das Tool StoryTec komplett benutzen zu können. |                          |                          |                          |                          |                          |                          |                          |          |
| _____ Stunden  |                          |                          |                          |                          |                          |                          |                          |          |

|  |  |
|--|--|
| <b>Welches der Tools würden Sie für die Lösung einer Aufgabe im Stil der von Ihnen während der Evaluation bearbeiteten Aufgaben verwenden?</b> |  |
|  | <input type="checkbox"/> E-Adventure <input type="checkbox"/> StoryTec |

|  |                          |
|--|--------------------------|
| <b>Wie lange arbeiten Sie (privat/beruflich) überhaupt schon mit ...</b>   |                          |
| Computern?   | _____ Jahre _____ Monate |
| anderen Autorenwerkzeugen?   | _____ Jahre _____ Monate |
| <b>Wie häufig arbeiten Sie (privat/beruflich) durchschnittlich mit ...</b> |                          |
| Computern?   | _____ Stunden pro Woche  |
| anderen Autorenwerkzeugen?   | _____ Stunden pro Woche  |

|                   |   |
|-------------------|---|
| Ihr Geschlecht?   | <input type="checkbox"/> weiblich <input type="checkbox"/> männlich |
| Wie alt sind Sie? | _____ Jahre   |

Vielen Dank für Ihre aktive Hilfe. Ihre Daten werden selbstverständlich anonym behandelt.  
Das letzte Wort haben Sie:

Bitte sehen Sie den Fragebogen nochmals durch, ob Sie alle Fragen beantwortet haben!

## B.4 COMPARISON STUDY TASK DESCRIPTION

**STORYTEC<sup>ST</sup>****Aufgabenstellung zur Evaluation**

Vielen Dank für Ihre Teilnahme an der Evaluation! Im Folgenden werden Sie mit einem Ausschnitt aus einem Serious Game aus dem Genre der Lern-Adventures arbeiten und verschiedene Teile des Spiels die entfernt wurden wieder herstellen. Sie werden die gleichen Aufgaben zunächst in StoryTec und dann in E-Adventure bearbeiten. Die Aufgaben sind identisch, nur die Hinweise und Tools unterscheiden sich.

Wenn Sie bei einer Aufgabe nicht mehr weiterkommen, speichern Sie Ihren Stand für diese Aufgabe ab und versuchen Sie die nächste Aufgabe.

**Beispiel-Video anschauen**

Um einen Eindruck vom fertigen Spiel zu bekommen, betrachten Sie das Video „Geograficus.avi“ das sie auf dem Desktop Ihres Rechners finden.

**Bearbeitung in StoryTec**

- Sie können Ihr Ergebnis jederzeit testen. Gehen Sie hierfür wie folgt vor: Speichern Sie das Spiel ab als Geograficus.icml in einem Ordner Ihrer Wahl und starten Sie es mit dem Player „StoryPlay“, den Sie auf dem Desktop Ihres Rechners finden.
- Orientieren Sie sich bei der Lösung an den noch vorhandenen Teilen des Spiels. Alle Elemente die sie hinzufügen sind in Ihrer Version des Spiels bereits enthalten.
- Benötigte Inhalte finden Sie auf dem Desktop Ihres Rechners in dem Ordner „Assets\StoryTec“.

**Aufgabe 1.1: Hintergrundbild ändern**

Starten Sie StoryTec über das Icon auf dem Desktop Ihres Rechners. Laden Sie die Datei „Spiel\StoryTec\Geograficus.icml“ auf dem Desktop.

In der Szene „tuer\_aussen\_zu“ wurde ein falsches Hintergrundbild eingefügt. Das richtige Hintergrundbild ist als „tuer\_aussen\_zu.jpg“ in dem „Assets“-Ordner zu finden. Ersetzen Sie das falsche Hintergrundbild mit dem Richtigen.

**Sichern**

Überprüfen Sie ob Sie die Aufgabenstellung komplett ausgeführt haben. Speichern Sie das Spiel ab, indem Sie im Hauptmenü den Punkt „Evaluation > Aufgabe 1 sichern“

**Aufgabe 1.2: Startszene wiederherstellen**

Die Startszene des Spiels, „schuppen\_aussen“, wurde entfernt. Stellen Sie diese Szene wieder her indem Sie eine neue Szene mit dem Namen „schuppen\_aussen“ erstellen. Diese Szene hat die folgenden Eigenschaften

- Das Hintergrundbild der Szene ist „schuppen\_aussen.jpg“ (wieder zu finden über den „Assets“-Ordner).
- Durch Klicken auf die Bildmitte wird in die Szene „tuer\_aussen\_zu“ gewechselt.
- Durch Klicken auf den unteren Bereich des Bildes wird in die Szene „aussen\_01“ gewechselt.

- Aus der Szene „aussen\_01“ gelangt man über Klicken auf den unteren Bereich des Bildes in die Szene.
- Das Spiel startet in dieser Szene.

## **Sichern**

Überprüfen Sie ob Sie die Aufgabenstellung komplett ausgeführt haben. Speichern Sie das Spiel ab, indem Sie im Hauptmenü den Punkt „Evaluation > Aufgabe 2 sichern“

## **Aufgabe 1.3: Rätsel vervollständigen**

In der Szene „edelsteine“ hat der Spieler die Aufgabe, eine Maschine zu bedienen und ein Muster aus Edelsteinen so zu konfigurieren daß eine vorgegebene Kombination entsteht. Sie finden die richtige Lösung auf dem Blatt an Ihrem Platz.

In Ihrer Version wurden die Edelsteine in der rechten unteren Ecke entfernt. Fügen Sie den Satz Edelsteine wieder ein und gehen Sie sicher daß das Rätsel im Spiel gelöst werden kann.

Stellen Sie hierbei sicher, daß

- die richtigen Grafiken für die Edelsteine in der rechten unteren Ecke angezeigt werden. Eine Liste der Grafiken finden Sie auf dem Zettel mit der Lösung.
- die Edelsteine beim Anklicken durchgewechselt werden.
- sobald das Rätsel durch das Einstellen der richtigen Kombination gelöst ist ein Sprung in die Szene „schuppen\_mit\_dose“ ausgeführt wird.

## **Sichern**

Überprüfen Sie ob Sie die Aufgabenstellung komplett ausgeführt haben. Speichern Sie das Spiel ab, indem Sie im Hauptmenü den Punkt „Evaluation > Aufgabe 3 sichern“

## Bearbeitung in E-Adventure

- Sie können Ihr Ergebnis jederzeit testen. Gehen Sie hierfür wie folgt vor: Wählen Sie den Menüpunkt „Run > Normal“ aus. Bitte verlassen Sie das Spiel immer über das Drücken der Escape-Taste und „Exit from game“.
- Orientieren Sie sich bei der Lösung an den noch vorhandenen Teilen des Spiels. Alle Elemente die sie hinzufügen sind in Ihrer Version des Spiels bereits enthalten.
- Benötigte Inhalte finden Sie auf dem Desktop Ihres Rechners in dem Ordner „Assets\E-Adventure“.

### Aufgabe 2.1: Hintergrundbild ändern

Starten Sie E-Adventure über das Icon auf dem Desktop Ihres Rechners. Laden Sie die Datei „Spiel\E-Adventure\Geograficus.eap“ auf dem Desktop.

In der Szene „tuer\_aussen\_zu“ wurde ein falsches Hintergrundbild eingefügt. Das richtige Hintergrundbild ist als „tuer\_aussen\_zu.jpg“ in dem „Assets“-Ordner zu finden. Ersetzen Sie das falsche Hintergrundbild mit dem Richtigen.

#### Sichern

Überprüfen Sie ob Sie die Aufgabenstellung komplett ausgeführt haben. Speichern Sie das Spiel ab, indem Sie im Hauptmenü den Punkt „Evaluation > Aufgabe 1 sichern“

### Aufgabe 2.2: Startszene wiederherstellen

Die Startszene des Spiels, „schuppen\_aussen“, wurde entfernt. Stellen Sie diese Szene wieder her indem Sie eine neue Szene mit dem Namen „schuppen\_aussen“ erstellen. Diese Szene hat die folgenden Eigenschaften

- Das Hintergrundbild der Szene ist „schuppen\_aussen.jpg“ (wieder zu finden über den „Assets“-Ordner).
- Durch Klicken auf die Bildmitte wird in die Szene „tuer\_aussen\_zu“ gewechselt.
- Durch Klicken auf den unteren Bereich des Bildes wird in die Szene „aussen\_01“ gewechselt.
- Aus der Szene „aussen\_01“ gelangt man über Klicken auf den unteren Bereich des Bildes in die Szene.
- Das Spiel startet in dieser Szene.

#### Sichern

Überprüfen Sie ob Sie die Aufgabenstellung komplett ausgeführt haben. Speichern Sie das Spiel ab, indem Sie im Hauptmenü den Punkt „Evaluation > Aufgabe 2 sichern“

### Aufgabe 2.3: Rätsel vervollständigen

In der Szene „edelsteine“ hat der Spieler die Aufgabe, eine Maschine zu bedienen und ein Muster aus Edelsteinen so zu konfigurieren daß eine vorgegebene Kombination entsteht. Sie finden die richtige Lösung auf dem Blatt an Ihrem Platz.

In Ihrer Version wurden die Edelsteine in der rechten unteren Ecke entfernt. Fügen Sie den Satz Edelsteine wieder ein und gehen Sie sicher daß das Rätsel im Spiel gelöst werden kann.



## AUTHOR'S PUBLICATIONS

---

### C.1 MAIN PUBLICATIONS

1. Florian Mehm, Stefan Göbel, and Ralf Steinmetz. *An Authoring Tool for Educational Adventure Games: Concept, Game Models and Authoring Processes*. *International Journal on Game-Based Learning*, 3(1):63–79, 2013.
2. Florian Mehm, Christian Reuter, and Stefan Göbel. *Authoring of Serious Games for Education*. In: Klaus Bredl and Wolfgang Bösch, editors, *Serious Games and Virtual Worlds in Education, Professional Development, and Healthcare*. IGI Global, 2013.
3. Florian Mehm, Stefan Göbel, and Ralf Steinmetz. *Authoring of Serious Adventure Games in StoryTec*. In: Stefan Göbel, Wolfgang Müller, Bodo Urban, and Josef Wiemeyer, editors, *E-Learning and Games for Training, Education, Health and Sports*. Springer, Berlin / Heidelberg, 2012.
4. Florian Mehm, Stefan Göbel, and Ralf Steinmetz. *Authoring and Re-Authoring Processes for Educational Adventure Games*. In: *Proceedings of the 6th European Conference on Games Based Learning*. Academic Conferences Limited, Reading, UK, 2012.
5. Florian Mehm, Johannes Konert, Stefan Göbel, and Ralf Steinmetz. *An Authoring Tool for Adaptive Digital Educational Games*. In: *Proceedings of the Seventh European Conference on Technology Enhanced Learning*. Springer, 2012.
6. Florian Mehm, Christian Reuter, Stefan Göbel, and Ralf Steinmetz. *Future Trends in Game Authoring Tools*. In: *Proceedings of the 2nd Workshop on Game Development and Model-Driven Software Development*. Springer, Berlin / Heidelberg, 2012.
7. Florian Mehm, Stefan Göbel, and Ralf Steinmetz. *Introducing Component-Based Templates into a Game Authoring Tool*. In: Dimitris Gouscos and Michalis Meimaris, editors, *5th European Conference on Games Based Learning*, pages 395–403. Academic Conferences Limited, Reading, UK, 2011. ISBN 978-1-908272-19-5 CD.
8. Florian Mehm, Sandro Hardy, Stefan Göbel, and Ralf Steinmetz. *Collaborative Authoring of Serious Games for Health*. In: *Proceedings of the 19th ACM international conference on Multimedia, MM '11*, pages 807–808. ACM, New York, NY, USA, 2011. ISBN 978-1-4503-0616-4.
9. Florian Mehm. *Authoring Serious Games*. In: Yusuf Pisan, editor, *Proceedings of Foundations of Digital Games 2010*, pages 271–273. ACM, New York, 2010.
10. Florian Mehm and Stefan Göbel. *Authoring-Tools für die Erstellung von Exergames*. In: Josef Wiemeyer, Daniel Link, Regine Angert, Bettina Holler, Annika Kliem,

- Nina Roznawski, Dietbert Schöberl, and Markus Stroß, editors, *Sportinformatik trifft Sporttechnologie: Abstractband zur Tagung der dvs-Sektion Sportinformatik und der deutschen interdisziplinären Vereinigung für Sporttechnologie*, pages 172–174. Institut für Sportwissenschaft der Technischen Universität Darmstadt, Darmstadt, 2010.
11. Florian Mehm, Stefan Göbel, and Ralf Steinmetz. *User Support in Digital Educational Game Authoring Tools*. In: Sybille Hambach, Alke Martens, Djamshid Tavangarian, and Bodo Urban, editors, *Proceedings of the 3rd International eLBA Science Conference*, pages 202–211. Fraunhofer Verlag, Stuttgart, 2010. ISBN 978-3-8396-0135-8.
  12. Florian Mehm, Viktor Wendel, Stefan Göbel, and Ralf Steinmetz. *Bat Cave: A Testing and Evaluation Platform for Digital Educational Games*. In: Bente Meyer, editor, *Proceedings of the 3rd European Conference on Games Based Learning*, pages 251–260. Academic Conferences International, Reading, UK, 2010.
  13. Florian Mehm, Viktor Wendel, Sabrina Radke, Stefan Göbel, Sebastian Grünwald, Robert Konrad, and Ralf Steinmetz. *Re-Authoring eines Lernadventures*. In: Holger Diener, Steffen Malo, Bodo Urban, Dennis Maciuszek, and Alke Martens, editors, *Spielend Lernen*, pages 27–42. Fraunhofer Verlag, Stuttgart, 2010. ISBN 978-3-8396-0186-0.
  14. Florian Mehm, Stefan Göbel, Sabrina Radke, and Ralf Steinmetz. *Authoring Environment for Story-based Digital Educational Games*. In: Michael D. Kickmeier-Rust, editor, *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*, pages 113–124. 2009.

## C.2 CO-AUTHORED PUBLICATIONS

15. Stefan Göbel and Florian Mehm. *Personalized, Adaptive Digital Educational Games using Narrative Game-based Learning Objects*. In: Klaus Bredl and Wolfgang Bösche, editors, *Serious Games and Virtual Worlds in Education, Professional Development, and Healthcare (in print)*. IGI Global, 2013.
16. Stefan Göbel, Florian Mehm, and Viktor Wendel. *Adaptive Digital Storytelling for Digital Educational Games*. In: Michael D. Kickmeier-Rust and Dietrich Albert, editors, *An Alien's Guide to Multi-Adaptive Educational Computer Games*, chapter 5, pages 89–104. Informing Science Press, Santa Rosa, USA, 2012. ISBN 1-932886-56-7.
17. Johannes Konert, Kristina Richter, Florian Mehm, Stefan Göbel, Regina Bruder, and Ralf Steinmetz. *Knowledge Sharing in the Classroom*. *Journal of Educational Technology & Society*, 2012.
18. Stefan Göbel, Sandro Hardy, Viktor Wendel, Florian Mehm, and Ralf Steinmetz. *Serious Games for Health - Personalized Exergames*. In: Alberto Del Bimbo, Shih-Fu Chang, and Arnold W. M. Smeulders, editors, *Proceedings of the 18th International Conference on Multimedia 2010*, pages 1663–1666. ACM, 2010.

19. Kevin Koidl, Florian Mehm, Cormac Hampson, Owen Conlan, and Stefan Göbel. *Dynamically adjusting Digital Educational Games towards Learning Objectives*. In: Bente Meyer, editor, *Proceedings of the 3rd European Conference on Games Based Learning*, pages 177–184. Reading, England, Academic Publishing Limited, 2010. ISBN 978-1-906638-79-5 CD.
20. Stefan Göbel, André de Carvalho Rodrigues, Florian Mehm, and Ralf Steinmetz. *Narrative Game-based Learning Objects for Story-based Digital Educational Games*. In: Michael D. Kickmeier-Rust, editor, *Proceedings of the 1st International Open Workshop on Intelligent Personalization and Adaptation in Digital Educational Games*, pages 43–53. 2009.
21. Stefan Göbel, Florian Mehm, Sabrina Radke, and Ralf Steinmetz. *80Days: Adaptive Digital Storytelling for Digital Educational Games*. In: Yiwei Cao, Anna Hannemann, Baltasar Fernández Manjón, Stefan Göbel, Cord Hockemeyer, and Emmanuel Stefanakis, editors, *Proceedings of the 2nd International Workshop on Story-Telling and Educational Games (STEG'09)*, no 498, 498. CEUR Workshop Proceedings, 2009.
22. Stefan Göbel, Luca Salvatore, and Robert Konrad. *StoryTec: A Digital Storytelling Platform for the Authoring and Experiencing of Interactive and Non-linear Stories*. In: Paolo Nesi Kia Ng and Jaime Delgado, editors, *Fourth International Conference on Automated Solutions for Cross Media Content and Multi-Channel Distribution*, DOI 10.1109, pages 103–110. IEEE computer society, 2008. ISBN 978-0-7695-3406-0.





CURRICULUM VITÆ

---

## PERSONAL INFORMATION

|                |              |
|----------------|--------------|
| Name           | Florian Mehm |
| Date of Birth  | May 29, 1982 |
| Place of Birth | Darmstadt    |
| Nationality    | German       |

## EDUCATION

|                 |   |
|-----------------|---|
| 01/2009–        | Technische Universität Darmstadt<br>Department of Electrical Engineering and<br>Information Technology<br>Research assistant at Multimedia Communications Lab (KOM) |
| 10/2002–12/2007 | Technische Universität Darmstadt<br>Department of Computer Science<br>Diplom Informatik   |
| 04/2001         | Gymnasium Viktoriaschule, Darmstadt<br>Abitur   |

## PROFESSIONAL EXPERIENCE

|                |   |
|----------------|---|
| 1/2008–11/2008 | Computer Graphics Center, Darmstadt<br>Research Assistant |
|----------------|---|

## TEACHING ACTIVITIES

|           |  |
|-----------|--|
| 2009–2012 | Tutor for 5 Bachelor's and Master's theses<br>(including "Diplomarbeiten")   |
| 2009–2012 | Contribution to lecture "Serious Games"  |
| 2009–2012 | Tutor for seminar "Digital Storytelling" and lab exercises<br>"Spielerische Edutainment-Anwendungen, Computerspiele,<br>Lernanwendungen, Storytelling" and "Game Technology" |

## SCIENTIFIC ACTIVITIES

|              |  |
|--------------|--|
| Organization | Foundations of Digital Games, Doctoral Consortium, Orga-<br>nization Committee, 2011 |
| Reviewer     | European Conference on Game-Based Learning, 2011-2013                                |

## SUPERVISED THESES

- Prusche, Lukas *Datenmodellierung und Kontrolle für Storytelling-basierte, adaptive Lernspiele*. Diplomarbeit, Technische Universität Darmstadt, August 2009. (Supervised in cooperation with Dr. Stefan Göbel)
- Gasparian, Eduard *Entwicklung von Methoden und Konzepten für ein mobiles, Storytelling-basiertes Lernspiel*. Diplomarbeit, Technische Universität Darmstadt, April 2010. (Supervised in cooperation with Dr. Stefan Göbel)
- Petzold, Christoph *Methoden und Konzepte für ein Autorensystem zur Erstellung digitaler Lern-Adventures*. Bachelor's Thesis, Technische Universität Darmstadt, September 2011.
- Buth, Deborah *Benutzerunterstützungsmechanismen in einem Autorensystem für digitale Lernspiele*. Bachelor's Thesis, Technische Universität Darmstadt, September 2012. (Supervised in cooperation with Michael Gutjahr)
- Becher, Jana *Konzeption und Umsetzung eines adaptiven Mathematik-Lernspiels*. Bachelor's Thesis, Technische Universität Darmstadt, May 2013 (planned). (Supervised in cooperation with Michael Gutjahr)

ERKLÄRUNG LAUT §9 DER PROMOTIONSORDNUNG

---

ICH versichere hiermit, dass ich die vorliegende Dissertation allein und nur unter Verwendung der angegebenen Literatur verfasst habe.

Die Arbeit hat bisher noch nicht zu Prüfungszwecken gedient.

*Darmstadt, 31. Januar 2013*

## COLOPHON

This document was typeset using the typographical look-and-feel `classicthesis` developed by André Miede. The style was inspired by Robert Bringhurst's seminal book on typography "*The Elements of Typographic Style*". `classicthesis` is available for both  $\text{\LaTeX}$  and  $\text{\LyX}$ :

<http://code.google.com/p/classicthesis/>

Happy users of `classicthesis` usually send a real postcard to the author, a collection of postcards received so far is featured here:

<http://postcards.miede.de/>